Machine learning manuscript No.
(will be inserted by the editor)

Logical reduction of metarules

Andrew Cropper · Sophie Tourret

the date of receipt and acceptance should be inserted later

Abstract Many forms of inductive logic programming (ILP) use *metarules*, second-order Horn clauses, to define the structure of learnable programs and thus the hypothesis space. Deciding which metarules to use for a given learning task is a major open problem and is a trade-off between efficiency and expressivity: the hypothesis space grows given more metarules, so we wish to use fewer metarules, but if we use too few metarules then we lose expressivity. In this paper, we study whether fragments of metarules can be logically reduced to minimal finite subsets. We consider two traditional forms of logical reduction: subsumption and entailment. We also consider a new reduction technique called *derivation reduction*, which is based on SLD-resolution. We compute reduced sets of metarules for fragments relevant to ILP and theoretically show whether these reduced sets are reductions for more general infinite fragments. We experimentally compare learning with reduced sets of metarules on three domains: Michalski trains, string transformations, and game rules. In general, derivation reduced sets of metarules outperforms subsumption and entailment reduced sets, both in terms of predictive accuracies and learning times.

1 Introduction

Many forms of inductive logic programming (ILP) [1, 5, 14, 19–21, 32, 33, 49, 54, 58, 62] use second-order Horn clauses, called *metarules*[1] as a form of declarative bias [53]. Metarules define the structure of learnable programs which in turn defines the hypothesis space. For instance, to learn the *grandparent/2* relation given the *parent/2* relation, the *chain* metarule would be suitable:

A. Cropper
University of Oxford, UK
E-mail: andrew.cropper@cs.ox.ac.uk

Sophie Tourret
Max Planck Institute for Informatics, Germany
E-mail: stourret@mpi-inf.mpg.de

[1] Metarules are also called *program schemata* [21], *second-order schemata* [54], and *clause templates* [1], amongst many other names.

$$P(A,B) \leftarrow Q(A,C),R(C,B)$$

In this metarule[2] the letters P, Q, and R denote existentially quantified second-order variables (variables that can be bound to predicate symbols) and the letters A, B and C denote universally quantified first-order variables (variables that can be bound to constant symbols). Given the *chain* metarule, the background *parent/2* relation, and examples of the *grandparent/2* relation, ILP approaches will try to find suitable substitutions for the existentially quantified second-order variables, such as the substitutions {P/grandparent, Q/parent, R/parent} to induce the theory:

$$grandparent(A,B) \leftarrow parent(A,C),parent(C,B)$$

However, despite the widespread use of metarules, there is little work determining which metarules to use for a given learning task. Instead, suitable metarules are assumed to be given as part of the background knowledge, and are often used without any theoretical justification. Deciding which metarules to use for a given learning task is a major open challenge [8, 10] and is a trade-off between efficiency and expressivity: the hypothesis space grows given more metarules [10, 38], so we wish to use fewer metarules, but if we use too few metarules then we lose expressivity. For instance, it is impossible to learn the *grandparent/2* relation using only metarules with monadic predicates.

In this paper, we study whether potentially infinite fragments of metarules can be logically reduced to minimal, or irreducible, finite subsets, where a fragment is a syntactically restricted subset of a logical theory [4].

Cropper and Muggleton [10] first studied this problem. They used Progol's entailment reduction algorithm [45] to identify entailment reduced sets of metarules, where a clause C is entailment redundant in a clausal theory $T \cup \{C\}$ when $T \models C$. To illustrate entailment redundancy, consider the following first-order clausal theory T_1, where p, q, r, and s are first-order predicates:

$$C_1 = p(A,B) \leftarrow q(A,B)$$
$$C_2 = p(A,B) \leftarrow q(A,B),r(A)$$
$$C_3 = p(A,B) \leftarrow q(A,B),r(A),s(B,C)$$

In T_1 the clauses C_2 and C_3 are entailment redundant because they are both logical consequences of C_1, i.e. $\{C_1\} \models \{C_2, C_3\}$. Because $\{C_1\}$ cannot be reduced, it is a minimal entailment reduction of T_1.

Cropper and Muggleton showed that in some cases as few as two metarules are sufficient to entail an infinite fragment of *chained*[3] second-order dyadic Datalog [10]. They also showed that learning with minimal sets of metarules improves predictive accuracies and reduces learning times compared to non-minimal sets. To illustrate how a finite subset of metarules could entail an infinite set, consider the set of metarules with only monadic literals and a single first-order variable A:

[2] The fully quantified rule is $\exists P \exists Q \exists R \forall A \forall B \forall C\ P(A,B) \leftarrow Q(A,C),R(C,B)$.

[3] A chained dyadic Datalog clause has the restriction that every first-order variable in a clause appears in exactly two literals and a path connects every literal in the body of C to the head of C. In other words, a chained dyadic Datalog clause has the form $P_0(X_0,X_1) \leftarrow P_1(X_0,X_2),P_2(X_2,X_3),\ldots,P_n(X_n,X_1)$ where the order of the arguments in the literals does not matter.

$$
\begin{aligned}
&M_1 = P(A) \leftarrow T_1(A)\\
&M_2 = P(A) \leftarrow T_1(A), T_2(A)\\
&M_3 = P(A) \leftarrow T_1(A), T_2(A), T_3(A)\\
&\dots\\
&M_n = P(A) \leftarrow T_1(A), T_2(A), \dots, T_n(A)\\
&\dots
\end{aligned}
$$

Although this set is infinite it can be entailment reduced to the single metarule M_1 because it implies the rest of the theory.

However, in this paper, we claim that entailment reduction is not always the most appropriate form of reduction. For instance, suppose you want to learn the *father/2* relation given the background relations *parent/2*, *male/1*, and *female/1*. Then a suitable hypothesis is:

$$father(A,B) \leftarrow parent(A,B), male(A)$$

To learn such a hypothesis one would need a metarule of the form $P(A,B) \leftarrow Q(A,B), R(A)$. Now suppose you have the metarules:

$$
\begin{aligned}
&M_1 = P(A,B) \leftarrow Q(A,B)\\
&M_2 = P(A,B) \leftarrow Q(A,B), R(A)
\end{aligned}
$$

Running entailment reduction on these metarules would remove M_2 because it is a logical consequence of M_1. However, it is impossible to learn the intended *father/2* relation given only M_1. As this example shows, entailment reduction can be too strong because it can remove metarules necessary to specialise a clause, where M_2 can be seen as a specialisation of M_1.

To address this issue, we introduce *derivation reduction*, a new form of reduction based on derivations, which we claim is a more suitable form of reduction for reducing sets of metarules. Let $\vdash$ represent derivability in SLD-resolution[4] [34], then a Horn clause C is derivationally redundant in a Horn theory $T \cup \{C\}$ when $T \vdash C$. A Horn theory is derivationally irreducible if it contains no derivationally redundant clauses. To illustrate the difference between entailment and derivation reduction, consider the metarules:

$$
\begin{aligned}
&M_1 = P(A,B) \leftarrow Q(A,B)\\
&M_2 = P(A,B) \leftarrow Q(A,B), R(A)\\
&M_3 = P(A,B) \leftarrow Q(A,B), R(A,B)\\
&M_4 = P(A,B) \leftarrow Q(A,B), R(A,B), S(A,B)
\end{aligned}
$$

Running entailment reduction on these metarules would result in the reduction $\{M_1\}$ because M_1 entails the rest of the theory. Likewise, running *subsumption reduction* [52] (described in detail in Section 3.5) would also result in the reduction $\{M_1\}$. By contrast, running derivation reduction would only remove M_4 because it can be derived by self-resolving M_3. The remaining metarules M_2 and M_3 are not derivationally redundant because there is no way to derive them from the other metarules.

[4] We use $\vdash$ to represent derivability of both first-order and second-order clauses. In practice we reason about second-order clauses using first-order resolution via encapsulation [10], which we describe in Section 3.3.

1.1 Contributions

In the rest of this paper, we study whether fragments of metarules relevant to ILP can be logically reduced to minimal finite subsets. We study three forms of reduction: subsumption [55], entailment [45], and derivation. We also study how learning with reduced sets of metarules affects learning performance. To do so, we supply Metagol [13], a meta-interpretive learning (MIL) [12, 48, 49] implementation, with different reduced sets of metarules and measure the resulting learning performance on three domains: Michalski trains [35], string transformations, and game rules [9]. In general, using derivation reduced sets of metarules outperforms using subsumption and entailment reduced sets, both in terms of predictive accuracies and learning times. Overall, our specific contributions are:

- We describe the logical reduction problem (Section 3).
- We describe subsumption and entailment reduction, and introduce derivation reduction, the problem of removing derivationally redundant clauses from a clausal theory (Section 3).
- We study the decidability of the three reduction problems and show, for instance, that the derivation reduction problem is undecidable for arbitrary Horn theories (Section 3).
- We introduce two general reduction algorithms that take a reduction relation as a parameter. We also study their complexity (Section 4).
- We run the reduction algorithms on finite sets of metarules to identify minimal sets (Section 5).
- We theoretically show whether infinite fragments of metarules can be logically reduced to finite sets (Section 5).
- We experimentally compare the learning performance of Metagol when supplied with reduced sets of metarules on three domains: Michalski trains, string transformations, and game rules (Section 6).

2 Related work

This section describes work related to this paper, mostly work on logical reduction techniques. We first, however, describe work related to MIL and metarules.

2.1 Meta-interpretive learning

Although the study of metarules has implications for many ILP approaches [1, 5, 14, 19–21, 32, 33, 49, 54, 58, 62], we focus on meta-interpretive learning (MIL), a form of ILP based on a Prolog meta-interpreter[5]. The key difference between a MIL learner and a standard Prolog meta-interpreter is that whereas a standard Prolog meta-interpreter attempts to prove a goal by repeatedly fetching first-order clauses whose heads unify with a given goal, a MIL learner additionally attempts to prove a goal by fetching second-order metarules, supplied as background knowledge (BK), whose heads unify with the goal. The resulting meta-substitutions are saved and can be reused in later proofs. Following the proof of a set of goals, a logic program is formed by projecting the meta-substitutions

[5] Although the MIL problem has also been encoded as an ASP problem [32].

onto their corresponding metarules, allowing for a form of ILP which supports predicate invention and learning recursive theories.

Most existing work on MIL has assumed suitable metarules as input to the problem, or has used metarules without any theoretical justification. In this paper, we try to address this issue by identifying minimal sets of metarules for interesting fragments of logic, such as Datalog, from which a MIL system can theoretically learn any logic program.

2.2 Metarules

McCarthy [43] and Lloyd [40] advocated using second-order logic to represent knowledge. Similarly, Muggleton et al. [47] argued that using second-order representations in ILP provides more flexible ways of representing BK compared to existing methods. Metarules are second-order Horn clauses and are used as a form of declarative bias [50, 53] to determine the structure of learnable programs which in turn defines the hypothesis space. In contrast to other forms of declarative bias, such as modes [45] or grammars [7], metarules are logical statements that can be reasoned about, such as to reason about the redundancy of sets of metarules, which we explore in this paper.

Metarules were introduced in the Blip system [19]. Kietz and Wrobel [33] studied generality measures for metarules in the RDT system. A generality order is necessary because the RDT system searches the hypothesis space (which is defined by the metarules) in a top-down general-to-specific order. A key difference between RDT and MIL is that whereas RDT requires metarules of increasing complexity (e.g. rules with an increasing number of literals in the body), MIL derives more complex metarules through SLD-resolution. This point is important because this ability allows MIL to start from smaller sets of primitive metarules. In this paper we try to identify such primitive sets.

Using metarules to build a logic program is similar to the use of refinement operators in ILP [51, 57] to build a definite clause literal-by-literal[6]. As with refinement operators, it seems reasonable to ask about completeness and irredundancy of a set of metarules, which we explore in this paper.

2.3 Logical redundancy

Detecting and eliminating redundancy in a clausal theory is useful in many areas of computer science. In ILP logically reducing a theory is useful to remove redundancy from a hypothesis space to improve learning performance [10, 22]. In general, simplifying or reducing a theory often makes a theory easier to understand and use, and may also have computational efficiency advantages.

2.3.1 Literal redundancy

Plotkin [52] used subsumption to decide whether a literal is redundant in a first-order clause. Joyner [31] independently investigated the same problem, which he called *clause condensation*, where a condensation of a clause C is a minimum cardinality subset C' of C such that $C' \models C$. Gottlob and Fermüller [26] improved Joyner's algorithm and also

[6] MIL uses example driven test-incorporation for finding consistent programs as opposed to the generate-and-test approach of clause refinement.

showed that determining whether a clause is condensed is coNP-complete. In contrast to removing redundant literals, we focus on removing redundant clauses.

2.3.2 Clause redundancy

Plotkin [52] introduced methods to decide whether a clause is subsumption redundant in a first-order clausal theory. This problem has also been extensively studied in the context of first-order logic with equality due to its application in superposition-based theorem proving [30, 63]. The same problem, and slight variants, has been extensively studied in the propositional case [36, 37]. Removing redundant clauses has numerous applications, such as to improve the efficiency of SAT [29]. In contrast to these works, we focus on reducing theories formed of second-order Horn clauses (without equality), which to our knowledge has not yet been extensively explored. Another difference is that we additionally study redundancy based on SLD-derivations.

Cropper and Muggleton [10] used Progol's entailment-reduction algorithm [45] to identify irreducible, or minimal, sets of metarules. Their approach removed entailment redundant clauses from sets of metarules. They identified theories that are (1) entailment complete for certain fragments of second-order Horn logic, and (2) minimal or irreducible in that no further reductions are possible. They demonstrated that in some cases as few as two clauses are sufficient to entail an infinite theory. However, they only considered small and highly constrained fragments of metarules. In particular, they focused on an *exactly-two-connected* fragment of metarules where each literal is dyadic and each first-order variable appears exactly twice in distinct literals. However, as discussed in the introduction, entailment reduction is not always the most appropriate form of reduction because it can remove metarules necessary to specialise a clause. Therefore, in this paper, we go beyond entailment reduction and introduce derivation reduction. We also consider more general fragments of metarules, such as a fragment of metarules sufficient to learn Datalog programs.

Cropper and Tourret [16] introduced the derivation reduction problem and studied whether sets of metarules could be derivationally reduced. They considered the *exactly-two-connected* fragment previously considered by Cropper and Muggleton and a *two-connected* fragment in which every variable appears at least twice, which is analogous to our singleton-free fragment (Section 5.3). They used graph theoretic methods to show that certain fragments could not be completely derivationally reduced. They demonstrated on the Michalski trains dataset that the partially derivationally reduced set of metarules outperforms the entailment reduced set. In similar work Cropper and Tourret elaborated on their graph theoretic techniques and expanded the results to unconstrained resolution [61].

In this paper, we go beyond the work of Cropper and Tourret in several ways. First, we consider more general fragments of metarules, including *connected* and *Datalog* fragments. We additionally consider fragments with zero arity literals. In all cases we provide additional theoretical results showing whether certain fragments can be reduced, and, where possible, show the actual reductions. Second, Cropper and Tourret [61] focused on derivation reduction modulo first-order variable unification, i.e. they considered the case where factorisation [51] was allowed when resolving two clauses, which is not implemented in practice in current MIL systems. For this reason, although Section 5 in [61] and Section 5.1 in the present paper seemingly consider the same problem, the results are opposite to one another. Third, in addition to entailment and derivation reduction, we also consider subsumption reduction. We provide more theoretical results on the

decidability of the reduction problems, such as showing a decidable case for derivation reduction (Theorem 4). Fourth, we describe the reduction algorithms and discuss their computational complexity. Finally, we corroborate the experimental results of Cropper and Tourret on Michalski's train problem [16] and provide additional experimental results on two more domains: real-world string transformations and inducing Datalog game rules from observations.

2.3.3 Theory minimisation

We focus on removing clauses from a clausal theory. A related yet distinct topic is theory minimisation where the goal is to find a minimum equivalent formula to a given input formula. This topic is often studied in propositional logic [28]. The minimisation problem allows for the introduction of new clauses. By contrast, the reduction problem studied in this paper does not allow for the introduction of new clauses and instead only allows for the removal of redundant clauses.

2.3.4 Prime implicates

Implicates of a theory T are the clauses that are entailed by T and are called prime when they do not themselves entail other implicates of T. This notion differs from the subsumption and derivation reduction because it focuses on entailment, and it differs from entailment reduction because (1) the notion of a prime implicate has been studied only in propositional, first-order, and some modal logics [2, 18, 42]; (2) the generation of prime implicates allows for the introduction of new clauses in the formula.

3 Logical reduction

We now introduce the reduction problem: the problem of finding redundant clauses in a theory. We first describe the reduction problem starting with preliminaries, and then describe three instances of the problem. The first two instances are based on existing logical reduction methods: subsumption and entailment. The third instance is a new form of reduction introduced in [16] based on SLD-derivations.

3.1 Preliminaries

We assume familiarity with logic programming notation [39] but we restate some key terminology. A *clause* is a disjunction of literals. A *clausal theory* is a set of clauses. A *Horn* clause is a clause with at most one positive literal. A Horn theory is a set of Horn clauses. A *definite* clause is a Horn clause with exactly one positive literal. A Horn clause is a *Datalog* clause if (1) it contains no function symbols, and (2) every variable that appears in the head of the clause also appears in a positive (i.e. not negated) literal in the body of the clause[7]. We denote the powerset of the set S as 2^S.

[7] Datalog also imposes additional constraints on negation in the body of a clause, but because we disallow negation in the body we omit these constraints for simplicity.

3.1.1 Metarules

Although the reduction problem applies to any clausal theory, we focus on theories formed of metarules:

Definition 1 (Metarule) A metarule is a second-order Horn clause of the form:

$$A_0 \leftarrow A_1, \ldots, A_m$$

where each A_i is a literal of the form $P(T_1, \ldots, T_n)$ where P is either a predicate symbol or a second-order variable that can be substituted by a predicate symbol, and each T_i is either a constant symbol or a first-order variable that can be substituted by a constant symbol.

Name	**Metarule**
Indent_1	$P(A) \leftarrow Q(A)$
DIndent_1	$P(A) \leftarrow Q(A), R(A)$
Indent_2	$P(A,B) \leftarrow Q(A,B)$
DIndent_2	$P(A,B) \leftarrow Q(A,B), R(A,B)$
Precon	$P(A,B) \leftarrow Q(A), R(A,B)$
Postcon	$P(A,B) \leftarrow Q(A,B), R(B)$
Curry	$P(A,B) \leftarrow Q(A,B,R)$
Chain	$P(A,B) \leftarrow Q(A,C), R(C,B)$

Table 1: Example metarules. The letters P, Q, and R denote existentially quantified second-order variables. The letters A, B, and C denote universally quantified first-order variables.

Table 1 shows a selection of metarules commonly used in the MIL literature [11, 12, 14, 15, 44]. As Definition 1 states, metarules may include predicate and constant symbols. However, we focus on the more general case where metarules only contain variables[8]. In addition, although metarules can be any Horn clauses, we focus on definite clauses with at least one body literal, i.e. we disallow facts, because their inclusion leads to uninteresting reductions, where in almost all such cases the theories can be reduced to a single fact[9] We denote the infinite set of all such metarules as $\mathcal{M}$. We focus on *fragments* of $\mathcal{M}$, where a fragment is a syntactically restricted subset of a theory [4]:

Definition 2 (The fragment $\mathcal{M}_m^a$) We denote as $\mathcal{M}_m^a$ the fragment of $\mathcal{M}$ where each literal has arity at most a and each clause has at most m literals in the body. We replace a by the explicit set of arities when we restrict the allowed arities further.

Example 1 $\mathcal{M}_2^{\{2\}}$ is a subset of $\mathcal{M}$ where each predicate has arity 2 and each clause has at most 2 body literals.

Example 2 $\mathcal{M}_m^{\{2\}}$ is a subset of $\mathcal{M}$ where each predicate has arity 2 and each clause has at most m body literals.

[8] By more general we mean we focus on metarules that are independent of any particular ILP problem with particular predicate and constant symbols.

[9] For instance, the metarule $P(A) \leftarrow$ entails and subsumes every metarule with a monadic head.

Example 3 $\mathcal{M}_m^{\{0,2\}}$ is a subset of $\mathcal{M}$ where each predicate has arity 0 or 2 and each clause has at most m body literals.

Example 4 $\mathcal{M}_{\{1,2\}}^a$ is a subset of $\mathcal{M}$ where each predicate has arity at most a and each clause has either 1 or 2 body literals.

Let T be a clausal theory. Then we say that T is in the fragment $\mathcal{M}_m^a$ if and only if each clause in T is in $\mathcal{M}_m^a$.

3.2 Meta-interpretive learning

In Section 6 we conduct experiments to see whether using reduced sets of metarules can improve learning performance. The primary purpose of the experiments is to test our claim that entailment reduction is not always the most appropriate form of reduction. Our experiments focus on MIL. For self-containment, we briefly describe MIL.

Definition 3 (MIL input) An MIL input is a tuple (B, E^+, E^-, M) where:

- B is a set of Horn clauses denoting background knowledge
- E^+ and E^- are disjoint sets of ground atoms representing positive and negative examples respectively
- M is a set of metarules

The MIL problem is defined from a MIL input:

Definition 4 (MIL problem) Given a MIL input (B, E^+, E^-, M), the MIL problem is to return a logic program hypothesis H such that:

- $\forall c \in H, \exists m \in M$ such that $c = m\theta$, where θ is a substitution that grounds all the existentially quantified variables in m
- $H \cup B \models E^+$
- $H \cup B \not\models E^-$

We call H a solution to the MIL problem.

The metarules and background define the hypothesis space. To explain our experimental results in Section 6, it is important to understand the effect that metarules have on the size of the MIL hypothesis space, and thus on learning performance. The following result generalises previous results [12, 38]:

Theorem 1 (MIL hypothesis space) *Given p predicate symbols and k metarules in $\mathcal{M}_m^a$, the number of programs expressible with n clauses is at most $(p^{m+1}k)^n$.*

Proof The number of first-order clauses which can be constructed from a $\mathcal{M}_m^a$ metarule given p predicate symbols is at most p^{m+1} because for a given metarule there are at most $m+1$ predicate variables with at most p^{m+1} possible substitutions. Therefore the set of such clauses S which can be formed from k distinct metarules in $\mathcal{M}_m^a$ using p predicate symbols has cardinality at most $p^{m+1}k$. It follows that the number of programs which can be formed from a selection of n clauses chosen from S is at most $(p^{m+1}k)^n$. □

Theorem 1 shows that the MIL hypothesis space increases given more metarules. The Blumer bound [3][10], says that given two hypothesis spaces, searching the smaller space will result in fewer errors compared to the larger space, assuming that the target hypothesis is in both spaces. This result suggests that we should consider removing redundant metarules to improve the learning performance. We explore this idea in the rest of the paper.

3.3 Encapsulation

To reason about metarules (especially when running the Prolog implementations of the reduction algorithms), we use a method called encapsulation [10] to transform a second-order logic program to a first-order logic program. We first define encapsulation for atoms:

Definition 5 (Atomic encapsulation) Let A be a second-order or first-order atom of the form $P(T_1,..,T_n)$. Then $enc(A) = enc(P,T_1,..,T_n)$ is the encapsulation of A.

For instance, the encapsulation of the atom *parent(ann,andy)* is *enc(parent,ann,andy)*. Note that encapsulation essentially ignores the quantification of variables in metarules by treating all variables, including predicate variables, as first-order universally quantified variables of the first-order *enc* predicate. In particular, replacing existential quantifiers with universal quantifiers on predicate variables is fine for our work because we only reason about the form of metarules, not their semantics, i.e. we treat metarules as templates for first-order clauses. We extend atomic encapsulation to logic programs:

Definition 6 (Program encapsulation) The logic program $enc(P)$ is the encapsulation of the logic program P in the case $enc(P)$ is formed by replacing all atoms A in P by $enc(A)$.

For example, the encapsulation of the metarule $P(A,B) \leftarrow Q(A,C),R(C,B)$ is $enc(P,A,B) \leftarrow enc(Q,A,C),enc(R,C,B)$. We extend encapsulation to interpretations [51] of logic programs:

Definition 7 (Interpretation encapsulation) Let I be an interpretation over the predicate and constant symbols in a logic program. Then the encapsulated interpretation $enc(I)$ is formed by replacing each atom A in I by $enc(A)$.

We now have the proposition:

Proposition 1 (Encapsulation models [10]) *The second-order logic program P has a model M if and only if $enc(P)$ has the model $enc(M)$.*

Proof Follows trivially from the definitions of encapsulated programs and interpretations. □

We can extend the definition of entailment to logic programs:

Proposition 2 (Entailment [10]) *Let P and Q be second-order logic programs. Then $P \models Q$ if and only if every model $enc(M)$ of $enc(P)$ is also a model of $enc(Q)$.*

[10] The Blumer bound is a reformulation of Lemma 2.1 in [3].

Proof Follows immediately from Proposition 1. □

These results allow us to reason about metarules using standard first-order logic. In the rest of the paper all the reasoning about second-order theories is performed at the first-order level. However, to aid the readability we continue to write non-encapsulated metarules in the rest of the paper, i.e. we will continue to refer to sets of metarules as second-order theories.

3.4 Logical reduction problem

We now describe the logical reduction problem. For the clarity of the paper, and to avoid repeating definitions for each form of reduction that we consider (entailment, subsumption, and derivability), we describe a general reduction problem which is parametrised by a binary relation $\sqsubset$ defined over any clausal theory, although in the case of derivability, $\sqsubset$ is in fact only defined over Horn clauses. Our only constraint on the relation $\sqsubset$ is that if $A \sqsubset B$, $A \subseteq A'$ and $B' \subseteq B$ then $A' \sqsubset B'$. We first define a redundant clause:

Definition 8 ($\sqsubset$-redundant clause) The clause C is $\sqsubset$-redundant in the clausal theory $T \cup \{C\}$ whenever $T \sqsubset \{C\}$.

In a slight abuse of notation, we allow Definition 8 to also refer to a single clause, i.e. in our notation $T \sqsubset C$ is the same as $T \sqsubset \{C\}$. We define a reduced theory:

Definition 9 ($\sqsubset$-reduced theory) A clausal theory is $\sqsubset$-reduced if and only if it is finite and it does not contain any $\sqsubset$-redundant clauses.

We define the input to the reduction problem:

Definition 10 ($\sqsubset$-reduction input) A reduction input is a pair (T,$\sqsubset$) where T is a clausal theory and $\sqsubset$ is a binary relation over a clausal theory.

Note that a reduction input may (and often will) be an infinite clausal theory. We define the reduction problem:

Definition 11 ($\sqsubset$-reduction problem) Let (T,$\sqsubset$) be a reduction input. Then the $\sqsubset$-reduction problem is to find a finite theory $T' \subseteq T$ such that (1) $T' \sqsubset T$ (i.e. $T' \sqsubset C$ for every clause C in T), and (2) T' is $\sqsubset$-reduced. We call T' a $\sqsubset$-*reduction*.

Although the input to a $\sqsubset$-reduction problem may contain an infinite theory, the output (a $\sqsubset$-*reduction*) must be a finite theory. We also introduce a variant of the $\sqsubset$-reduction problem where the reduction must obey certain syntactic restrictions:

Definition 12 ($\mathscr{M}_m^a$-$\sqsubset$-reduction problem) Let (T,$\sqsubset$,$\mathscr{M}_m^a$) be a triple, where the first two elements are as in a standard reduction input and $\mathscr{M}_m^a$ is a target reduction theory. Then the $\mathscr{M}_m^a$-$\sqsubset$-reduction problem is to find a finite theory $T' \subseteq T$ such that (1) T' is a $\sqsubset$-reduction of T, and (2) T' is in $\mathscr{M}_m^a$.

3.5 Subsumption reduction

The first form of reduction we consider is based on subsumption, which, as discussed in Section 2, is often used to eliminate redundancy in a clausal theory:

Definition 13 (Subsumption) A clause C subsumes a clause D, denoted as $C \preceq D$, if there exists a substitution θ such that $C\theta \subseteq D$.

Note that if a clause C subsumes a clause D then $C \models D$ [55]. However, if $C \models D$ then it does not necessarily follow that $C \preceq D$. Subsumption can therefore be seen as being weaker than entailment. Whereas checking entailment between clauses is undecidable [6], Robinson [55] showed that checking subsumption between clauses is decidable (although in general deciding subsumption is a NP-complete problem [51]).

If T is a clausal theory then the pair $(T, \preceq)$ is an input to the $\sqsubset$-reduction problem, which leads to the *subsumption reduction* problem (S-reduction problem). We show that the S-reduction problem is decidable for finite theories:

Proposition 3 (Finite S-reduction problem decidability) *Let T be a finite theory. Then the corresponding S-reduction problem is decidable.*

Proof We can enumerate each element T' of 2^T in ascending order on the cardinality of T'. For each T' we can check whether T' subsumes T, which is decidable because subsumption between clauses is decidable. If T' subsumes T then we correctly return T'; otherwise we continue to enumerate. Because the set 2^T is finite the enumeration must halt. Because the set 2^T contains T the algorithm will in the worst-case return T. Thus the problem is decidable. □

3.6 Entailment reduction

As mentioned in the introduction, Cropper and Muggleton [10] previously used entailment reduction [45] to reduce sets of metarules using the notion of an entailment redundant clause:

Definition 14 (E-redundant clause) The clause C is entailment redundant (E-redundant) in the clausal theory $T \cup \{C\}$ whenever $T \models C$.

If T is a clausal theory then the pair $(T, \models)$ is an input to the $\sqsubset$-reduction problem, which leads to the entailment reduction problem (E-reduction). We show the relationship between an E- and a S-reduction:

Proposition 4 *Let T be a clausal theory, T_S be a S-reduction of T, and T_E be an E-reduction of T. Then $T_E \models T_S$.*

Proof Assume the opposite, i.e. $T_E \not\models T_S$. This assumption implies that there is a clause $C \in T_S$ such that $T_E \not\models C$. By the definition of S-reduction, T_S is a subset of T so C must be in T, which implies that $T_E \not\models T$. But this contradicts the premise that T_E is an E-reduction of T. Therefore the assumption cannot hold, and thus $T_E \models T_S$. □

We show that the E-reduction problem is undecidable for arbitrary clausal theories:

Proposition 5 (E-reduction problem clausal decidability) *The E-reduction problem for clausal theories is undecidable.*

Proof Follows from the undecidability of entailment in clausal logic [6]. □

The E-reduction problem for Horn theories is also undecidable:

Proposition 6 (E-reduction problem Horn decidability) *The E-reduction problem for Horn theories is undecidable.*

Proof Follows from the undecidability of entailment in Horn logic [41]. □

The E-reduction problem is, however, decidable for finite Datalog theories:

Proposition 7 (E-reduction problem Datalog decidability) *The E-reduction problem for finite Datalog theories is decidable.*

Proof Follows from the decidability of entailment in Datalog [17] using a similar algorithm to the one used in the proof of Proposition 3. □

3.7 Derivation reduction

As mentioned in the introduction, entailment reduction can be too strong a form of reduction. We therefore describe a new form of reduction based on derivability [16, 61]. Although our notion of derivation reduction can be defined for any proof system (such as unconstrained resolution as is done in [61]) we focus on SLD-resolution because we want to reduce sets of metarules, which are definite clauses. We define the function $R^n(T)$ of a Horn theory T as:

$R^0(T) = T$
$R^n(T) = \{C | C_1 \in R^{n-1}(T), C_2 \in T, C \text{ is the binary resolvent of } C_1 \text{ and } C_2\}$

We use this function to define the Horn closure of a Horn theory:

Definition 15 (Horn closure) The Horn closure $R^*(T)$ of a Horn theory T is:

$$\bigcup_{n \in \mathbb{N}} R^n(T)$$

We state our notion of derivability:

Definition 16 (Derivability) A Horn clause C is derivable from the Horn theory T, written $T \vdash C$, if and only if $C \in R^*(T)$.

We define a *derivationally redundant* (D-redundant) clause:

Definition 17 (D-redundant clause) A clause C is derivationally redundant in the Horn theory $T \cup \{C\}$ if $T \vdash C$.

Let T be a Horn theory, then the pair $(T, \vdash)$ is an input to the $\sqsubset$-reduction problem, which leads to the *derivation reduction* problem (D-reduction problem). Note that a theory can have multiple D-reductions. For instance, consider the theory T:

$$C_1 = P(A,B) \leftarrow Q(B,A)$$
$$C_2 = P(A,B) \leftarrow Q(A,C), R(C,B)$$
$$C_3 = P(A,B) \leftarrow Q(C,A), R(C,B)$$

One D-reduction of T is $\{C_1, C_2\}$ because we can resolve the first body literal of C_2 with C_1 to derive C_3 (up to variable renaming). Another D-reduction of T is $\{C_1, C_3\}$ because we can likewise resolve the first body literal of C_3 with C_1 to derive C_2.

We can show the relationship between E- and D-reductions by restating the notion of a SLD-deduction [51]:

Definition 18 (SLD-deduction [51]) Let T be a Horn theory and C be a Horn clause. Then there exists a SLD-deduction of C from T, written $T \vdash_d C$, if C is a tautology or if there exists a clause D such that $T \vdash D$ and D subsumes C.

We can use the *subsumption theorem* [51] to show the relationship between SLD-deductions and logical entailment:

Theorem 2 (SLD-subsumption theorem [51]) *Let T be a Horn theory and C be a Horn clause. Then $T \models C$ if and only if $T \vdash_d C$.*

We can use this result to show the relationship between an E- and a D-reduction:

Proposition 8 *Let T be a Horn theory, T_E be an E-reduction of T, and T_D be a D-reduction of T. Then $T_E \models T_D$.*

Proof Follows from the definitions of E-reduction and D-reduction because an E-reduction can be obtained from a D-reduction with an additional subsumption check. □

We also use the SLD-subsumption theorem to show that the D-reduction problem is undecidable for Horn theories:

Theorem 3 (D-reduction problem Horn decidability) *The D-reduction problem for Horn theories is undecidable.*

Proof Assume the opposite, that the problem is decidable, which implies that $T \vdash C$ is decidable. Since $T \vdash C$ is decidable and subsumption between Horn clauses is decidable [24], then finding a SLD-deduction is also decidable. Therefore, by the SLD-subsumption theorem, entailment between Horn clauses is decidable. However, entailment between Horn clauses is undecidable [56], so the assumption cannot hold. Therefore, the problem must be undecidable. □

However, the D-reduction problem is decidable for any fragment $\mathscr{M}^a_m$ (e.g. definite Datalog clauses where each clause has at least one body literal, with additional arity and body size constraints). To show this result, we first introduce two lemmas:

Lemma 1 *Let D, C_1, and C_2 be definite clauses with m_d, m_{c1}, and m_{c2} body literals respectively, where m_d, m_{c1}, and $m_{c2} > 0$. If $\{C_1, C_2\} \vdash D$ then $m_{c1} \leq m_d$ and $m_{c2} \leq m_d$.*

Proof Follows from the definitions of SLD-resolution [51]. □

Note that Lemma 1 does not hold for unconstrained resolution because it allows for factorisation [51]. Lemma 1 also does not hold when facts (bodyless definite clauses) are allowed because they would allow for resolvents that are smaller in body size than one of the original two clauses.

Lemma 2 *Let $\mathcal{M}_m^a$ be a fragment of metarules. Then $\mathcal{M}_m^a$ is finite up to variable renaming.*

Proof Any literal in $\mathcal{M}_m^a$ has at most a first-order variables and 1 second-order variable, so any literal has at most $a+1$ variables. Any metarule has at most m body literals plus the head literal, so any metarule has at most $m+1$ literals. Therefore, any metarule has at most $((a+1)(m+1))$ variables. We can arrange the variables in at most $((a+1)(m+1))!$ ways, so there are at most $((a+1)(m+1))!$ metarules in $\mathcal{M}_m^a$ up to variable renaming. Thus $\mathcal{M}_m^a$ is finite up to variable renaming. □

Note that the bound in the proof of Lemma 2 is a worst-case result. In practice there are fewer usable metarules because we consider fragments of constrained theories, thus not all clauses are admissible, and in all cases the order of the body literals is irrelevant. We use these two lemmas to show that the D-reduction problem is decidable for $\mathcal{M}_m^a$:

Theorem 4 ($\mathcal{M}_m^a$-D-reduction problem decidability) *The D-reduction problem for theories included in $\mathcal{M}_m^a$ is decidable.*

Proof Let T be a finite clausal theory in $\mathcal{M}_m^a$ and C be a definite clause with $n > 0$ body literals. The problem is whether $T \vdash C$ is decidable. By Lemma 1, we cannot derive C from any clause which has more than n body literals. We can therefore restrict the resolution closure $R^*(T)$ to only include clauses with body lengths less than or equal to n. In addition, by Lemma 2 there are only a finite number of such clauses so we can compute the fixed-point of $R^*(T)$ restricted to clauses of size smaller or equal to n in a finite amount of steps and check whether C is in the set. If it is then $T \vdash C$; otherwise $T \nvdash C$. □

3.8 k-derivable clauses

Propositions 3 and 7 and Theorem 4 show that the $\sqsubset$-reduction problem is decidable under certain conditions. However, as we will shown in Section 4, even in decidable cases, solving the $\sqsubset$-reduction problem is computationally expensive. We therefore solve restricted k-bounded versions of the E- and D-reduction problems, which both rely on SLD-derivations. Specifically, we focus on resolution depth-limited derivations using the notion of *k-derivability*:

Definition 19 (k-derivability) Let k be a natural number. Then a Horn clause C is k-derivable from the Horn theory T, written $T \vdash_k C$, if and only if $C \in R^k(T)$.

The definitions for k-bounded E- and D-reductions follow from this definition but are omitted for brevity. In Section 4 we introduce a general algorithm (Algorithm 1) to solve the S-reduction problem and k-bounded E- and D-reduction problems.

4 Reduction algorithms

In Section 5 we logically reduce sets of metarules. We now describe the reduction algorithms that we use.

4.1 $\sqsubset$-reduction algorithm

The reduce algorithm (Algorithm 1) shows a general $\sqsubset$-reduction algorithm that solves the $\sqsubset$-reduction problem (Definition 11) when the input theory is finite[11]. We ignore cases where the input is infinite because of the inherent undecidability of the problem. Algorithm 1 is largely based on Plotkin's clausal reduction algorithm [52]. Given a finite clausal theory T and a binary relation $\sqsubset$, the algorithm repeatedly tries to remove a $\sqsubset$-redundant clause in T. If it cannot find a $\sqsubset$-redundant clause, then it returns the $\sqsubset$-reduced theory. Note that since derivation reduction is only defined over Horn theories, in a $\vdash$-reduction input $(T, \vdash)$, the theory T has to be Horn. We show total correctness of the algorithm:

Proposition 9 (Algorithm 1 total correctness) *Let $(T, \sqsubset)$ be a $\sqsubset$-reduction input where T is finite. Let the corresponding $\sqsubset$-reduction problem be decidable. Then Algorithm 1 solves the $\sqsubset$-reduction problem.*

Proof Trivial by induction on the size of T. □

Algorithm 1 $\sqsubset$-reduction algorithm

```
1  func reduce(T,⊏):
2      if |T| < 2:
3          return T
4      if there is a clause C in T such that T\{C} ⊏ C:
5          return reduce(T\{C},⊏)
6      else:
7          return T
```

Note that Proposition 9 assumes that the given reduction problem is decidable and that the input theory is finite. If you call Algorithm 1 with an arbitrary clausal theory and the $\models$ relation then it will not necessarily terminate. We can call Algorithm 1 with specific binary relations, where each variation has a different time-complexity. Table 2 shows different ways of calling Algorithm 1 with their corresponding time complexities, where we assume finite theories as input. We show the complexity of calling Algorithm 1 with the subsumption relation:

Proposition 10 (S-reduction complexity) *If T is a finite clausal theory then calling Algorithm 1 with $(T, \preceq)$ requires at most $O(|T|^3)$ calls to a subsumption algorithm.*

Proof For every clause in T the algorithm checks whether any other clause in T subsumes C which requires at most $O(|T|^2)$ calls to a subsumption algorithm. If any clause C is found to be S-redundant then the algorithm repeats the procedure on the theory $(T \setminus \{C\})$, so overall the algorithm requires at most $O(|T|^3)$ calls to a subsumption algorithm. □

[11] In practice we use more efficient algorithms for each approach. For instance, in the derivation reduction Prolog implementation we use the knowledge gained from Lemma 1 to add pruning so as to ignore clauses that are too large to be useful to check whether a clause is derivable.

Relation	Output	Complexity
$\preceq$	S-reduction	$O(\|T\|^3)$
$\models$	E-reduction	Undecidable
$\models_k$	k-E-reduction	$O(\|T\|^{k+2})$
$\vdash$	D-reduction	Undecidable
$\vdash_k$	k-D-reduction	$O(\|T\|^{k+2})$

Table 2: Outputs and complexity of Algorithm 1 for different input relations and an arbitrary finite clausal theory T. The time complexities are a function of the size of the given theory, denoted by $|T|$.

Note that a more detailed analysis of calling Algorithm 1 with the subsumption relation would depend on the subsumption algorithm used, which is an NP-complete problem [24]. We show the complexity of calling Algorithm 1 with the k-bounded entailment relation:

Proposition 11 (k-bounded E-reduction complexity) *If T is a finite Horn theory and k is a natural number then calling Algorithm 1 with (T,$\models_k$) requires at most $O(|T|^{k+2})$ resolutions.*

Proof In the worst case the derivation check (line 4) requires searching the whole SLD-tree which has a maximum branching factor $|T|$ and a maximum depth k and takes $O(|T|^k)$ steps. The algorithm potentially does this step for every clause in T so the complexity of this step is $O(|T|^{k+1})$. The algorithm has to perform this check for every clause in T with an overall worst-case complexity $O(|T|^{k+2})$. □

The complexity of calling Algorithm 1 with the k-derivation relation is identical:

Proposition 12 (k-bounded D-reduction complexity) *Let T be a finite Horn theory and k be a natural number then calling Algorithm 1 with (T,$\vdash_k$) requires at most $O(|T|^{k+2})$ resolutions.*

Proof Follows using the same reasoning as Proposition 11. □

4.2 $\mathscr{M}^a_m$-$\sqsubset$-reduction algorithm

Although Algorithm 1 solves the $\sqsubset$-reduction problem, it does not solve the $\mathscr{M}^a_m$-reduction problem (Definition 12). For instance, suppose you have the following theory T in $\mathscr{M}^2_4$:

$$M_1 = P(A,B) \leftarrow Q(B,A)$$
$$M_2 = P(A,B) \leftarrow Q(A,A), R(B,B)$$
$$M_3 = P(A,B) \leftarrow Q(A,C), R(B,C)$$
$$M_4 = P(A,B) \leftarrow Q(B,C), R(A,D), S(A,D), T(B,C)$$

Suppose you want to know whether T can be E-reduced to $\mathcal{M}_2^2$. Then calling Algorithm 1 with $(T, \models)$ (i.e. the entailment relation) will return $T' = \{M_1, M_4\}$ because: $M_4 \models M_2$[12], $M_4 \models M_3$[13], and $\{M_1, M_4\}$ cannot be further E-reduced.

Although T' is an E-reduction of T, it is not in $\mathcal{M}_2^2$ because M_4 is not in $\mathcal{M}_2^2$. However, the theory T can be $\mathcal{M}_2^2$-E-reduced to $\{M_1, M_2, M_3\}$ because $\{M_2, M_3\} \models M_4$[14], and $\{M_1, M_2, M_3\}$ cannot be further reduced. In general, let T be a theory in $\mathcal{M}_m^a$ and an T' be an E-reduction of T, then T' is not necessarily in $\mathcal{M}_2^a$.

Algorithm 2 overcomes this limitation of Algorithm 1. Given a finite clausal theory T, a binary relation $\sqsubset$, and a reduction fragment $\mathcal{M}_m^a$, Algorithm 2 determines whether there is a $\sqsubset$-reduction of T in $\mathcal{M}_m^a$. If there is, it returns the reduced theory; otherwise it returns false. In other words, Algorithm 2 solves the $\mathcal{M}_m^a$-$\sqsubset$-reduction problem. We show total correctness of Algorithm 2:

Proposition 13 (Algorithm 2 correctness) *Let $(T, \sqsubset, \mathcal{M}_m^a)$ be a $\mathcal{M}_m^a$-$\sqsubset$-reduction input. If the corresponding $\sqsubset$-reduction problem is decidable then Algorithm 2 solves the corresponding $\mathcal{M}_m^a$-$\sqsubset$-reduction problem.*

Sketch Proof We provide a sketch proof for brevity. We need to show that the function aux correctly determines whether $B \sqsubset T$, which we can show by induction on the size of T. Assuming aux is correct, then if T can be reduced to B, the mreduce function calls Algorithm 1 to reduce B, which is correct by Proposition 9. Otherwise it returns false. □

Algorithm 2 $\mathcal{M}_m^a$-$\sqsubset$-reduction

```
1  function mreduce(T,⊏,M_m^a)
2      B = {C|C ∈ T ∩ M_m^a}
3      if aux(T,⊏,B):
4          return reduce(B,⊏)
5      return false
6
7  function aux(T,⊏,B)
8      if |T| == 0:
9          return true
10     pick any C in T
11     T' = T \ {C}
12     if B ⊏ C:
13         return aux(T',⊏,B)
14     return false
```

[12] Rename the variables in M_4 to form $M_4' = P_0(X_1,X_2) \leftarrow P_1(X_2,X_3), P_2(X_1,X_4), P_3(X_1,X_4), P_4(X_2,X_3)$. Then $M_4'\theta = P(A,B) \leftarrow R(B,B), Q(A,A), Q(A,A), R(B,B)$ where $\theta = \{P_0/P, P_1/R, P_2/Q, P_3/Q, P_4/R, X_1/A, X_2/B, X_3/B, X_4/A\}$. It follows that $M_4'\theta \subseteq M_2$, so $M_4 \preceq M_2$, which in turn implies $M_4 \models M_2$.

[13] Rename the variables in M_4 to form $M_4' = P_0(X_1,X_2) \leftarrow P_1(X_2,X_3), P_2(X_1,X_4), P_3(X_1,X_4), P_4(X_2,X_3)$. Then $M_4'\theta = P(A,B) \leftarrow R(B,C), Q(A,C), Q(A,C), R(B,C)$ where $\theta = \{P_0/P, P_1/R, P_2/Q, P_3/Q, P_4/R, X_1/A, X_2/B, X_3/C, X_4/C\}$. It follows that $M_4'\theta \subseteq M_3$, so $M_4 \preceq M_3$, which in turn implies $M_4 \models M_3$.

[14] Rename the variables in M_3 to form $M_3' = P_0(X,Y) \leftarrow P_1(X,Z), P_2(Y,Z)$. Resolve the first body literal of M_2 with M_3 to form $R_1 = P(A,B) \leftarrow P_1(A,Z), P_2(A,Z), R(B,B)$. Rename the variables P_1 to P_3, P_2 to P_4, and Z to Z_1 in R_1 (to standardise apart the variables) to form $R_2 = P(A,B) \leftarrow P_3(A,Z_1), P_4(A,Z_1), R(B,B)$. Resolve the last body literal of R_2 with M_3' to form $R_3 = P(A,B) \leftarrow P_3(A,Z_1), P_4(A,Z_1), P_1(B,Z), P_2(B,Z)$. Rename the variables Z_1 to D, $ZtoC$, P_3 to R, P_4 to S, P_1 to Q, and P_2 to T in R_3 to form $R_4 = P(A,B) \leftarrow R(A,D), S(A,D), Q(B,C), T(B,C)$.Thus, $R_4 = M_4$, so it follows that and $\{M_2, M_3\} \models M_4$

Fragment	Description
$\mathscr{C}$	connected clauses
$\mathscr{D}$	connected Datalog clauses
$\mathscr{K}$	connected Datalog clauses without singleton variables
$\mathscr{U}$	connected Datalog clauses without duplicate variables

Table 3: The four main fragments of $\mathscr{M}$ that we consider.

5 Reduction of metarules

We now logically reduce fragments of metarules. Given a fragment $\mathscr{M}^a_m$ and a reduction operator $\sqsubset$, we have three main goals:

G1: identify a $\mathscr{M}^a_k$-$\sqsubset$-reduction of $\mathscr{M}^a_m$ for some k as small as possible
G2: determine whether $\mathscr{M}^a_2 \sqsubset \mathscr{M}^a_\infty$
G3: determine whether $\mathscr{M}^a_\infty$ has any (finite) $\sqsubset$-reduction

We work on these goals for fragments of $\mathscr{M}^a_m$ relevant to ILP. Table 3 shows the four fragments and their main restrictions. The subsequent sections precisely describe the fragments.

Our first goal (**G1**) is to essentially minimise the number of body literals in a set of metarules, which can be seen as trying to enforce an Occamist bias. We are particularly interested reducing sets of metarules to fragments with at most two body literals because $\mathscr{M}^{\{2\}}_2$ augmented with one function symbol has universal Turing machine expressivity [60]. In addition, previous work on MIL has almost exclusively used metarules from the fragment $\mathscr{M}^2_2$. Our second goal (**G2**) is more general and concerns reducing an infinite set of metarules to $\mathscr{M}^a_2$. Our third goal (**G3**) is similar, but is about determining whether an infinite set of metarules has any finite reduction.

We work on the goals by first applying the reduction algorithms described in the previous section to finite fragments restricted to 5 body literals (i.e. $\mathscr{M}^a_5$). This value gives us a sufficiently large set of metarules to reduce but not too large that the reduction problem is intractable. When running the E- and D-reduction algorithms (both k-bounded), we use a resolution-depth bound of 7, which is the largest value for which the algorithms terminate in reasonable time[15]. After applying the reduction algorithms to the finite fragments, we then try to solve **G2** by extrapolating the results to the infinite case (i.e. $\mathscr{M}^a_\infty$). In cases where $\mathscr{M}^a_2 \not\sqsubset \mathscr{M}^a_\infty$, we then try to solve **G3** by seeing whether there exists any natural number k such that $\mathscr{M}^a_k \sqsubset \mathscr{M}^a_\infty$.

5.1 Connected ($\mathscr{C}^a_m$) results

We first consider a general fragment of metarules. The only constraint is that we follow the standard ILP convention [10, 20, 27, 51] and focus on connected clauses[16]:

[15] The entailment and derivation reduction algorithms often took 4-5 hours to find a reduction. However, in some cases, typically where the fragments contained many metarules, the algorithms took around 12 hours to find a reduction. By contrast, the subsumption reduction algorithm typically found a reduction in 30 minutes.

[16] Connected clauses are also known as linked clauses [27].

Definition 20 (Connected clause) A clause is connected if the literals in the clause cannot be partitioned into two sets such that the variables appearing in the literals of one set are disjoint from the variables appearing in the literals of the other set.

The following clauses are all connected:

$$P(A) \leftarrow Q(A)$$
$$P(A,B) \leftarrow Q(A,C)$$
$$P(A,B) \leftarrow Q(A,B), R(B,D), S(D,B)$$

By contrast, these clauses are not connected:

$$P(A) \leftarrow Q(B)$$
$$P(A,B) \leftarrow Q(A), R(C)$$
$$P(A,B) \leftarrow Q(A,B), S(C)$$

We denote the connected fragment of $\mathscr{M}_m^a$ as $\mathscr{C}_m^a$. Table 4 shows the maximum body size and the cardinality of the reductions obtained when applying the reduction algorithms to $\mathscr{C}_5^a$ for different values of a. To give an idea of the scale of the reductions, the fragment $\mathscr{C}_5^{\{1,2\}}$ contains 77398 unique metarules, of which E-reduction removed all but two of them. Table 5 shows the actual reductions for $\mathscr{C}_5^{\{1,2\}}$. Reductions for other connected fragments are in Appendix A.1.

Arities	S-reduction		E-reduction		D-reduction	
a	Bodysize	Cardinality	Bodysize	Cardinality	Bodysize	Cardinality
0	1	1	1	1	2	2
1	1	1	1	1	2	2
2	1	4	1	1	5	6
0,1	1	3	1	3	2	5
0,2	1	6	1	3	5	21
1,2	1	9	1	2	4	8
0,1,2	1	12	1	4	5	13

Table 4: Cardinality and maximal body size of the reductions of $\mathscr{C}_5^a$. All the fragments can be S- and E-reduced to $\mathscr{C}_1^a$ but they cannot all be D-reduced to $\mathscr{C}_2^a$.

S-reduction	E-reduction	D-reduction
$P(A) \leftarrow Q(A)$	$P(A) \leftarrow Q(B,A)$	$P(A) \leftarrow Q(B,A)$
$P(A) \leftarrow Q(A,B)$	$P(A,B) \leftarrow Q(A)$	$P(A,A) \leftarrow Q(B,A)$
$P(A) \leftarrow Q(B,A)$		$P(A,B) \leftarrow Q(B)$
$P(A,B) \leftarrow Q(A)$		$P(A,B) \leftarrow Q(B,A)$
$P(A,B) \leftarrow Q(B)$		$P(A,B) \leftarrow Q(B,B)$
$P(A,B) \leftarrow Q(A,C)$		$P(A,B) \leftarrow Q(A,B), R(A,B)$
$P(A,B) \leftarrow Q(B,C)$		$P(A,B) \leftarrow Q(A,C), R(B,C)$
$P(A,B) \leftarrow Q(C,A)$		$P(A,B) \leftarrow Q(A,C), R(A,D), S(B,C), T(B,D), U(C,D)$
$P(A,B) \leftarrow Q(C,B)$		

Table 5: Reductions of the connected fragment $\mathscr{C}_5^{\{1,2\}}$.

As Table 4 shows, all the fragments can be S- and E-reduced to $\mathscr{C}_1^a$. We show that in general $\mathscr{C}_\infty^a$ has a $\mathscr{C}_1^a$-S-reduction:

Theorem 5 **($\mathscr{C}^a_\infty$ S-reducibility)** *For all $a > 0$, the fragment $\mathscr{C}^a_\infty$ has a $\mathscr{C}^a_1$-S-reduction.*

Proof Let C be any clause in $\mathscr{C}^a_\infty$, where $a > 0$. By the definition of connected clauses there must be at least one body literal in C that shares a variable with the head literal of C. The clause formed of the head of C with the body literal directly connected to it is by definition in $\mathscr{C}^a_1$ and clearly subsumes C. Therefore $\mathscr{C}^a_1 \preceq \mathscr{C}^a_\infty$. □

We likewise show that $\mathscr{C}^a_\infty$ always has a $\mathscr{C}^a_1$-E-reduction:

Theorem 6 **($\mathscr{C}^a_\infty$ E-reducibility)** *For all $a > 0$, the fragment $\mathscr{C}^a_\infty$ has a $\mathscr{C}^a_1$-E-reduction.*

Proof Follows from Theorem 5 and Proposition 4. □

As Table 4 shows, the fragment $\mathscr{C}^2_5$ could not be D-reduced to $\mathscr{C}^2_2$ when running the derivation reduction algorithm. However, because we run the derivation reduction algorithm with a maximum derivation depth, this result alone is not enough to guarantee that the output cannot be further reduced. Therefore, we show that $\mathscr{C}^2_5$ cannot be D-reduced to $\mathscr{C}^2_2$:

Proposition 14 **($\mathscr{C}^2_5$ D-irreducibility)** *The fragment $\mathscr{C}^2_5$ has no $\mathscr{C}^2_2$-D-reduction.*

Proof We denote by $\mathscr{P}(C)$ the set of all clauses that can be obtained from a given clause C by permuting the arguments in its literals up to variable renaming. For example if $C = P(A,B) \leftarrow Q(A,C)$ then $\mathscr{P}(C) = \{(C), (P(A,B) \leftarrow Q(C,A)), (P(B,A) \leftarrow Q(A,C)), (P(B,A) \leftarrow Q(C,A))\}$ up to variable renaming.

Let C_I denote the clause $P(A,B) \leftarrow Q(A,C), R(A,D), S(B,C), T(B,D), U(C,D)$. We prove that no clause in $\mathscr{P}(C_I)$ can be derived from $\mathscr{C}^2_2$ by induction on the length of derivations. Formally, we show that there exist no derivations of length n from $\mathscr{C}^2_2$ to a clause in $\mathscr{P}(C_I)$. We reason by contradiction and w.l.o.g. we consider only the clause C_I.

For the base case $n = 0$, assume that there is a derivation of length 0 from $\mathscr{C}^2_2$ to C_I. This assumption implies that $C_I \in \mathscr{C}^2_2$, but this clearly cannot hold given the body size of C_I.

For the general case, assume that the property holds for all $k < n$ and by contradiction consider the final inference in a derivation of length n of C_I from $\mathscr{C}^2_2$. Let C_1 and C_2 denote the premises of this inference. Then the literals occurring in C_I must occur up to variable renaming in at least one of C_1 and C_2. We consider the following cases separately.

- All the literals of C_I occur in the same premise: because of Lemma 1, this case is impossible because this premise would contain more literals than C_I (the ones from C_I plus the resolved literal).
- Only one of the literals of C_I occurs separately from the others: w.l.o.g., assume that the literal $Q(A,C)$ occurs alone in C_2 (up to variable renaming). Then C_2 must be of the form $H(A,C) \leftarrow Q(A,C)$ or $H(C,A) \leftarrow Q(A,C)$ for some H, where the H-headed literal is the resolved literal of the inference that allows the unification of A and C with their counterparts in C_1[17]. In this case, C_1 belongs to $\mathscr{P}(C_I)$ and a derivation of C_1 from $\mathscr{C}^2_2$ of length smaller than n exists as a strict subset of the derivation to C_I of length n. This contradicts the induction hypothesis, thus the assumed derivation of C cannot exist.

[17] Those are the only options to derive C_I.Otherwise, e.g. with $C_2 = H(A',C') \leftarrow Q(A',D')$, the resulting clause is not C_I because D' is not unified with any of the variables in C_1 (whereas A' unifies with A and C' with C), e.g. the result includes the literal $Q(A,D')$ instead of $Q(A,C)$ hence it is not C_I.

– Otherwise, the split of the literals of C_I between C_1 and C_2 is always such that at least three variables must be unified during the inference. For example, consider the case where $P(A,B) \leftarrow Q(A,C) \subset C_1$ and the set $\{R(A',D), S(B',C'), T(B',D), U(C',D)\}$ occurs in the body of C_2 (up to variable renaming). Then A', B' and C' must unify respectively with A, B and C for C_I to be derived (up to variable renaming). However the inference can at most unify two variable pairs since the resolved literal must be dyadic at most and thus this inference is impossible, a contradiction.

Thus C_I and all of $\mathscr{P}(C_I)$ cannot be derived from $\mathscr{C}_2^2$. Note that, since $\mathscr{P}(C_I)$ is also neither a subset of $\mathscr{C}_3^2$ nor of $\mathscr{C}_4^2$, this proof also shows that $\mathscr{P}(C_I)$ cannot be derived from $\mathscr{C}_3^2$ and from $\mathscr{C}_4^2$. □

We generalise this result to $\mathscr{C}_\infty^2$:

Theorem 7 ($\mathscr{C}_\infty^2$ D-irreducibility) *The fragment $\mathscr{C}_\infty^2$ has no D-reduction.*

Proof It is enough to prove that $\mathscr{C}_\infty^2$ does not have a $\mathscr{C}_m^2$-D-reduction for an arbitrary m because any D-reduced theory, being finite, admits a bound on the body size of the clauses it contains. Starting from C_I as defined in the proof of Proposition 14, apply the following transformation iteratively for k from 1 to m: replace the literals containing Q and R (i.e. at first $Q(A,C)$ and $R(A,D)$) with the following set of literals $Q(A,C_k)$, $R(A,D_k)$, $V_k(C_k,D_k)$, $Q_k(C_k,C)$, $R_k(D_k,D)$ where all variables and predicate variables labeled with k are new. Let the resulting clause be denoted C_{I_m}. This clause is of body size $3m+5$ and thus does not belong to $\mathscr{C}_m^2$. Moreover, for the same reason that C_I cannot be derived from any $\mathscr{C}_{m'}^2$ with $m' < 5$ (see the proof of Proposition 14) C_{I_m} cannot be derived from any $\mathscr{C}_{m'}^2$ with $m' < 3m+5$. In particular, C_{I_m} cannot be derived from $\mathscr{C}_m^2$. □

Another way to generalise Proposition 14 is the following:

Theorem 8 ($\mathscr{C}_\infty^a$ D-irreducibility) *For $a \geq 2$, the fragment $\mathscr{C}_\infty^a$ has no $\mathscr{C}_{a^2+a-2}^a$-D-reduction.*

Proof Let C_a denote the clause

$$\begin{aligned} C_a = P(A_1,\dots,A_a) \leftarrow & Q_{1,1}(A_1,B_{1,1},\dots,B_{1,a-1})\dots Q_{1,a}(A_1,B_{a,1},\dots,B_{a,a-1}) \\ & \dots \\ & Q_{a,1}(A_a,B_{1,1},\dots,B_{1,a-1})\dots Q_{a,a}(A_a,B_{a,1},\dots,B_{a,a-1}) \\ & R_1(B_{1,1},\dots,B_{a,1}),\dots,R_{a-1}(B_{1,a-1},\dots,B_{a,a-1}) \end{aligned}$$

Note that for $a = 2$, the clauses C_a and C_I from the proof of Proposition 14 coincide. In fact to show that C_a is irreducible for any a, it is enough to consider the proof of Proposition 14 where C_a is substituted to C_I and where the last case is generalised in the following way:

– the split of the literals of C_a between C_1 and C_2 is always such that at least $a+1$ variables must be unified during the inference, which is impossible since the resolved literal can at most hold a variables.

The reason this proof holds is that any subset of C_a contains at least $a+1$ distinct variables. Since C_a is of body size a^2+a-1, this counter-example proves that $\mathscr{C}_\infty^a$ has no $\mathscr{C}_{a^2+a-2}^a$-D-reduction. □

Note that this is enough to conclude that $\mathscr{C}_\infty^a$ cannot be reduced to $\mathscr{C}_2^a$ but it does not prove that $\mathscr{C}_\infty^a$ is not D-reducible.

5.1.1 Summary

Table 6 summarises our theoretical results from this section. Theorems 5 and 6 show that $\mathscr{C}^a_\infty$ can always be S- and E-reduced to $\mathscr{C}^a_1$ respectively. By contrast, Theorem 7 shows that $\mathscr{C}^2_\infty$ cannot be D-reduced to $\mathscr{C}^2_2$. In fact, Theorem 7 says that $\mathscr{C}^2_\infty$ has no D-reduction. Theorem 7 has direct (negative) implications for MIL systems such as Metagol and HEXMIL. We discuss these implications in more detail in Section 7.

Arity	S	E	D
1	✓	✓	✓
2	✓	✓	×
>2	✓	✓	×

Table 6: Existence of a S-, E- or D-reduction of $\mathscr{C}^a_\infty$ to $\mathscr{C}^a_2$. The symbol ✓ denotes that the fragment does have a reduction. The symbol × denotes that the fragment does not have a reduction.

5.2 Datalog ($\mathscr{D}^a_m$) results

We now consider Datalog clauses, which are often used in ILP [1, 12, 20, 32, 49, 58]. The relevant Datalog restriction is that if a variable appears in the head of a clause then it must also appear in a body literal. If we look at the S-reductions of $\mathscr{C}^{\{1,2\}}_5$ in Table 5 then the clause $P(A,B) \leftarrow Q(B)$ is not a Datalog clause because the variable A appears in the head but not in the body. We denote the Datalog fragment of $\mathscr{C}^a_m$ as $\mathscr{D}^a_m$. Table 7 shows the results of applying the reduction algorithms to $\mathscr{D}^a_5$ for different values of a. Table 8 shows the reductions for the fragment $\mathscr{D}^{\{1,2\}}_5$, which are used in Experiment 3 (Section 6.3) to induce Datalog game rules from observations. Reductions for other Datalog fragments are in Appendix A.2.

Arities	S-reduction		E-reduction		D-reduction	
a	Bodysize	Cardinality	Bodysize	Cardinality	Bodysize	Cardinality
0	1	1	1	1	2	2
1	1	1	1	1	2	2
2	2	4	2	2	5	10
0,1	1	2	1	2	2	5
0,2	2	5	2	3	5	38
1,2	2	10	2	3	5	11
0,1,2	2	11	2	4	5	14

Table 7: Cardinality and maximal body size of the reductions of $\mathscr{D}^a_5$. All the fragments can be S- and E-reduced to $\mathscr{D}^a_2$ but they cannot all be D-reduced to $\mathscr{D}^a_2$.

We show that $\mathscr{D}^2_\infty$ can be S-reduced to $\mathscr{D}^2_2$:

Proposition 15 ($\mathscr{D}^2_\infty$ S-reducibility) *The fragment $\mathscr{D}^2_\infty$ has a $\mathscr{D}^2_2$-S-reduction.*

S-reduction	E-reduction	D-reduction
$P(A) \leftarrow Q(A)$	$P(A) \leftarrow Q(A,B)$	$P(A) \leftarrow Q(B,A)$
$P(A) \leftarrow Q(A,B)$	$P(A,B) \leftarrow Q(B,A)$	$P(A,A) \leftarrow Q(A)$
$P(A) \leftarrow Q(B,A)$	$P(A,B) \leftarrow Q(A),R(B)$	$P(A,A) \leftarrow Q(A,A)$
$P(A,A) \leftarrow Q(B,A)$		$P(A,B) \leftarrow Q(B,A)$
$P(A,B) \leftarrow Q(A),R(B)$		$P(A,B) \leftarrow Q(A,B),R(A,B)$
$P(A,B) \leftarrow Q(A),R(B,C)$		$P(A,B) \leftarrow Q(A,C),R(B,C)$
$P(A,B) \leftarrow Q(A,B)$		$P(A,B) \leftarrow Q(B,C),R(A,D)$
$P(A,B) \leftarrow Q(B),R(A,C)$		$P(A,B) \leftarrow Q(B,C),R(A,D),S(B,D),T(C,E)$
$P(A,B) \leftarrow Q(B,A)$		$P(A,B) \leftarrow Q(A,C),R(A,D),S(B,C),T(B,D),U(C,D)$
$P(A,B) \leftarrow Q(B,C),R(A,D)$		$P(A,B) \leftarrow Q(B,C),R(A,D),S(C,E),T(B,F),U(D,F)$
		$P(A,B) \leftarrow Q(B,C),R(B,D),S(C,E),T(A,F),U(D,F)$

Table 8: Reductions of the Datalog fragment $\mathscr{D}_5^{\{1,2\}}$.

Proof Follows using the same argument as in Theorem 5 but the reduction is to $\mathscr{D}_2^2$ instead of $\mathscr{D}_1^2$. This difference is due to the Datalog constraint that states: if a variable appears in the head it must also appear in the body. For clauses with dyadic heads, if the two head argument variables occur in two distinct body literals then the clause cannot be further reduced beyond $\mathscr{D}_2^2$. □

We show how this result cannot be generalised to $\mathscr{D}_\infty^a$:

Theorem 9 ($\mathscr{D}_\infty^a$ S-irreducibility) *For $a > 0$, the fragment $\mathscr{D}_\infty^a$ does not have a $\mathscr{D}_{a-1}^a$-S-reduction.*

Proof As a counter-example to a $\mathscr{D}_{a-1}^a$-S-reduction, consider $C_a = P(X_1,\ldots,X_a) \leftarrow Q_1(X_1),\ldots,Q_a(X_a)$. The clause C_a does not belong to $\mathscr{D}_{a-1}^a$ and cannot be S-reduced to it because the removal of any subset of its literals leaves argument variables in the head without their counterparts in the body. Hence, any subset of C_a does not belong to the Datalog fragment. Thus C_a cannot be subsumed by a clause in $\mathscr{D}_{a-1}^a$. □

However, we can show that $\mathscr{D}_\infty^a$ can always be S-reduced to $\mathscr{D}_a^a$:

Theorem 10 ($\mathscr{D}_\infty^a$ to $\mathscr{D}_a^a$ S-reducibility) *For $a > 0$, the fragment $\mathscr{D}_\infty^a$ has a $\mathscr{D}_a^a$-S-reduction.*

Proof To prove that $\mathscr{D}_\infty^a$ has a $\mathscr{D}_a^a$-S-reduction it is enough to remark that any clause in $\mathscr{D}_\infty^a$ has a subclause of body size at most a that is also in $\mathscr{D}_\infty^a$, the worst case being clauses such as C_a where all argument variables in the head occur in a distinct literal in the body. □

We also show that $\mathscr{D}_\infty^a$ always has a $\mathscr{D}_2^a$-E-reduction, starting with the following lemma:

Lemma 3 *For $a > 0$ and $n \in \{1,\ldots,a\}$, the clause*

$$P_0(A_1,A_2,\ldots,A_n) \leftarrow P_1(A_1),P_2(A_2),\ldots,P_n(A_n)$$

is $\mathscr{D}_2^a$-E-reducible.

Proof By induction on n.

– For the base case $n = 2$, by definition $\mathscr{D}_2^a$ contains $P_0(A_1,A_2) \leftarrow P_1(A_1),P_2(A_2)$

– For the inductive step, assume the claim holds for $n-1$. We show it holds for n. By definition $\mathscr{D}_2^a$ contains the clause $D_1 = P(A_1,A_2,\ldots,A_n) \leftarrow P_0(A_1,A_2,\ldots,A_{n-1}),P_n(A_n)$. By the inductive hypothesis, $D_2 = P_0(A_1,A_2,\ldots,A_{n-1}) \leftarrow P_1(A_1),\ldots,P_{n-1}(A_{n-1})$ is $\mathscr{D}_2^{a-1}$-E-reducible, and thus also $\mathscr{D}_2^a$-E-reducible. Together, D_1 and D_2 entail $D = P_0(A_1,A_2,\ldots,A_n) \leftarrow P_1(A_1),P_2(A_2),\ldots,P_n(A_n)$, which can be seen by resolving the literal $P_0(A_1,A_2,\ldots,A_{n-1})$ from D_1 with the same literal from D_2 to derive D. Thus D is $\mathscr{D}_2^a$-E-reducible.

□

Theorem 11 ($\mathscr{D}_\infty^a$ E-reducibility) *For $a > 0$, the fragment $\mathscr{D}_\infty^a$ has a $\mathscr{D}_2^a$-E-reduction.*

Proof Let C be any clause in $\mathscr{D}_\infty^a$. We denote the head of C by $P(A_1,\ldots,A_n)$, where $0 < n \leq a$. The possibility that some of the A_i are equal does not impact the reasoning.

If $n = 1$, then by definition, there exists a literal L_1 in the body of C such that A_1 occurs in L_1. It is enough to consider the clause $P(A_1) \leftarrow L_1$ to conclude, because $P(A_1)$ is the head of C and L_1 belongs to the body of C, thus $P(A_1) \leftarrow L_1$ entails C, and this clause belongs to $\mathscr{D}_2^a$.

In the case where $n > 1$, there must exist literals $L_1,\ldots,L_n$ in the body of C such that A_i occurs in L_i for $i \in \{1,\ldots,n\}$. Consider the clause $C' = P(A_1,\ldots,A_n) \leftarrow L_1,\ldots,L_n$. There are a few things to stress about C':

– The clause C' belongs to $\mathscr{D}_\infty^a$.
– Some L_i may be identical with each other, since the A_is may occur together in literals or simply be equal, but this scenario does not impact the reasoning.
– The clause C' entails C because C' is equivalent to a subset of C (but this subset may be distinct from C' due to C' possibly including some extra duplicated literals).

Now consider the clause $D = P(A_1,\ldots,A_n) \leftarrow P_1(A_1),\ldots,P_n(A_n)$. For $i \in \{1,\ldots,n\}$, the clause $P_i(A_i) \leftarrow L_i$ belongs to $\mathscr{D}_2^a$ by definition, thus $\mathscr{D}_2^a \cup D \vdash D'$ where $D' = P(A_1,\ldots,A_n) \leftarrow L_1,\ldots,L_n$. Moreover, by Lemma 3, D is $\mathscr{D}_2^a$-E-reducible, hence D' is also $\mathscr{D}_2^a$-E-reducible. Note that this notation hides the fact that if a variable occurs in distinct body literals L_i in C', this connection is not captured in D' where distinct variables will occur instead, thus there is no guarantee that $D' = C'$. For example, if $C' = P(A_1,A_2) \leftarrow Q(A_1,B,A_2),R(A_2,B)$ then $D' = P(A_1,A_2) \leftarrow Q(A_1,B,A'_2),Q(A_1,B,A'_2),R(A_2,B'),R(A_2,B')$ However, it always holds that $D' \models C'$, because D' subsumes C'. In our small example, it is enough to consider the substitution $\theta = \{B'/B,A'_2/A_2\}$ to observe this. Thus by transitivity of entailment, we can conclude that C is $\mathscr{D}_2^a$-E-reducible.

□

As Table 7 shows, not all of the fragments can be D-reduced to $\mathscr{D}_2^a$. In particular, the result that $\mathscr{D}_\infty^2$ has no $\mathscr{D}_2^2$-D-reduction follows from Theorem 7 because the counterexamples presented in the proof also belong to $\mathscr{D}_\infty^2$.

5.2.1 Summary

Table 9 summarises our theoretical results from this section. Theorem 9 shows that $\mathscr{D}_\infty^a$ never has a $\mathscr{D}_{a-1}^a$-S-reduction. This result differs from the connected fragment where $\mathscr{C}_\infty^a$ could always be S-reduced to $\mathscr{C}_2^a$. However, Theorem 9 shows that $\mathscr{D}_\infty^a$ can always be S-reduced to $\mathscr{D}_a^a$. As with the connected fragment, Theorem 11 shows that $\mathscr{D}_\infty^a$ can always be E-reduced to $\mathscr{C}_2^a$. The result that $\mathscr{D}_\infty^2$ has no D-reduction follows from Theorem 7.

Arity	S	E	D
1	✓	✓	✓
2	✓	✓	×
>2	×	✓	×

Table 9: Existence of a S-, E- or D-reduction of $\mathscr{D}^a_\infty$ to $\mathscr{D}^a_2$.

5.3 Singleton-free ($\mathscr{K}^a_m$) results

It is common in ILP to require that all the variables in a clause appear at least twice [10, 46, 54], which essentially eliminates singleton variables. We call this fragment the *singleton-free* fragment:

Definition 21 (Singleton-free) A clause is singleton-free if each first-order variable appears at least twice

For example, if we look at the E-reductions of the connected fragment $\mathscr{C}^{\{1,2\}}_5$ shown in Table 5 then the clause $P(A) \leftarrow Q(B,A)$ is not singleton-free because the variable B only appears once. We denote the singleton-free fragment of $\mathscr{D}^a_m$ as $\mathscr{K}^a_m$. Table 10 shows the results of applying the reduction algorithms to $\mathscr{K}^a_5$. Table 11 shows the reductions of $\mathscr{K}^{\{2\}}_5$. Reductions for other singleton-free fragments are in Appendix A.3.

Arities	S-reduction		E-reduction		D-reduction	
a	Bodysize	Cardinality	Bodysize	Cardinality	Bodysize	Cardinality
0	1	1	1	1	2	2
1	1	1	1	1	2	2
2	4	3	2	3	5	7
0,1	1	2	1	2	2	5
0,2	5	4	2	3	5	23
1,2	4	8	2	4	5	8
0,1,2	4	9	2	5	5	11

Table 10: Cardinality and maximal body size of the reductions of $\mathscr{K}^a_5$.

S-reduction	E-reduction	D-reduction
$P(A,B) \leftarrow Q(A,B)$	$P(A,B) \leftarrow Q(B,A)$	$P(A,A) \leftarrow Q(A,A)$
$P(A,B) \leftarrow Q(B,A)$	$P(A,B) \leftarrow Q(A,A),R(B,B)$	$P(A,B) \leftarrow Q(B,A)$
$P(A,B) \leftarrow Q(B,C),R(A,D),$	$P(A,B) \leftarrow Q(A,C),R(B,C)$	$P(A,A) \leftarrow Q(A,B),R(B,B)$
$S(A,D),T(B,C)$		$P(A,B) \leftarrow Q(A,A),R(B,B)$
		$P(A,B) \leftarrow Q(A,B),R(A,B)$
		$P(A,B) \leftarrow Q(A,C),R(B,C)$
		$P(A,B) \leftarrow Q(A,C),R(A,D),$
		$S(B,C),T(B,D),U(C,D)$

Table 11: Reductions of the singleton-free fragment $\mathscr{K}^{\{2\}}_5$

Unlike in the connected and Datalog cases, the fragment $\mathscr{K}^{\{2\}}_5$ is no longer S-reducible to $\mathscr{K}^{\{2\}}_2$. We show that $\mathscr{K}^2_\infty$ cannot be reduced to $\mathscr{K}^2_2$.

Proposition 16 ($\mathcal{K}^2_\infty$ S-reducibility) *The fragment $\mathcal{K}^2_\infty$ does not have a $\mathcal{K}^2_2$-S-reduction.*

Proof As a counter-example, consider the clause:

$$C = P(A,B) \leftarrow Q(A,D), R(A,D), S(B,C), T(B,C)$$

Consider removing any non-empty subset of literals from the body of C. Doing so leads to a singleton variable in the remaining clause, so it is not a singleton-free clause. Moreover, for any other clause to subsume C it must be more general than C, but that is not possible again because of the singleton-free constraint[18]. □

We can likewise show that this result holds in the general case:

Theorem 12 ($\mathcal{K}^a_\infty$ S-reducibility) *For $a \geq 2$, the fragment $\mathcal{K}^a_\infty$ does not have a $\mathcal{K}^a_{2a-1}$-S-reduction.*

Proof We generalise the clause C from the proof of Proposition 16 to define the clause $C_a = P(A_1,\ldots,A_a) \leftarrow P_1(A_1,B_1), P_2(A_1,B_1),\ldots,P_{2a-1}(A_a,B_a), P_{2a}(A_a,B_a)$. The same reasoning applies to C_a as to $C(=C_2)$, making C_a irreducible in $\mathcal{K}^a_\infty$. Moreover C_a is of body size $2a$, thus C_a is a counterexample to a $\mathcal{K}^a_{2a-1}$-S-reduction of $\mathcal{K}^a_\infty$. □

However, all the fragments can be E-reduced to $\mathcal{K}^a_2$.

Theorem 13 ($\mathcal{K}^a_\infty$ E-reducibility) *For $a > 0$, the fragment $\mathcal{K}^a_\infty$ has a $\mathcal{K}^a_2$-E-reduction.*

Proof The proof of Theorem 13 is an adaptation of that of Theorem 11. The only difference is that if $n = 1$ then $P(A_1) \leftarrow L_1, L_1$ must be considered instead of $P(A_1) \leftarrow L_1$ to ensure the absence of singleton variables in the body of the clause, and for the same reason, in the general case, the clause $D' = P(A_1,\ldots,A_n) \leftarrow L_1,...,L_n$ must be replaced by $D' = P(A_1,\ldots,A_n) \leftarrow L_1, L_1,\ldots,L_n, L_n$. Note that C' is not modified and thus may or may not belong to $\mathcal{K}^a_\infty$. However, it is enough that $C' \in \mathcal{D}^a_\infty$. With these modifications, the proof carries from $\mathcal{K}^a_\infty$ to $\mathcal{K}^a_2$ as from $\mathcal{D}^a_\infty$ to $\mathcal{D}^a_2$, including the results in Lemma 3. □

5.3.1 Summary

Table 12 summarises our theoretical results from this section. Theorem 12 shows that for $a \geq 2$, the fragment $\mathcal{K}^a_\infty$ does not have a $\mathcal{K}^a_{2a-1}$-S-reduction. This result contrasts with the Datalog fragment where $\mathcal{D}^a_\infty$ always has a $\mathcal{D}^a_a$-S-reduction. As is becoming clear, adding more restrictions to a fragment typically results in less S-reducibility. By contrast, as with the connected and Datalog fragments, Theorem 13 shows that fragment $\mathcal{K}^a_\infty$ always has a $\mathcal{K}^a_2$-E-reduction. In addition, as with the other fragments, $\mathcal{K}^a_\infty$ has no D-reduction for $a \geq 2$.

[18] Note that this proof also shows that $\mathcal{K}^2_\infty$ does not have a $\mathcal{K}^2_3$-S-reduction.

Arity	S	E	D
1	✓	✓	✓
2	×	✓	×
>2	×	✓	×

Table 12: Existence of a S-, E- or D-reduction of $\mathscr{K}^a_\infty$ to $\mathscr{K}^a_2$.

5.4 Duplicate-free ($\mathscr{U}^a_m$) results

The previous three fragments are general in the sense that they have been widely used in ILP. By contrast, the final fragment that we consider is of particular interest to MIL. Table 1 shows a selection of metarules commonly used in the MIL literature. These metarules have been successfully used despite no theoretical justification. However, if we consider the reductions of the three fragments so far, the *identity*, *precon*, and *postcon* metarules do not appear in any reduction. These metarules can be derived from the reductions, typically using either the $P(A) \leftarrow Q(A,A)$ or $P(A,A) \leftarrow Q(A)$ metarules. To try to identify a reduction which more closely matches the metarules shown in Table 1, we consider a fragment that excludes clauses in which a literal contains multiple occurrences of the same variable. For instance, this fragment excludes the previously mentioned metarules and also excludes the metarule $P(A,A) \leftarrow Q(B,A)$, which was in the D-reduction shown in Table 5. We call this fragment *duplicate-free*. It is a sub-fragment of $\mathscr{K}^a_m$ and we denote it as $\mathscr{U}^a_m$.

Table 13 shows the reductions for the fragment $\mathscr{U}^{\{1,2\}}_5$. Reductions for other duplicate-free fragments are in Appendix A.4. As Table 13 shows, the D-reduction of $\mathscr{U}^{\{1,2\}}_5$ contains some metarules commonly used in the MIL literature. For instance, it contains the $identity_1$, $didentity_2$, and *precon* metarules. We use the metarules shown in Table 13 in Experiments 1 and 2 (Sections 6.1 and 6.2) to learn Michalski trains solutions and string transformation programs respectively.

Table 14 shows the results of applying the reduction algorithms to $\mathscr{U}^a_5$ for different values of a. All the theoretical results that hold for the singleton-free fragments hold similarly for the duplicate-free fragments for the following reasons:

- (S) The clauses in the proofs of Proposition 16 and Theorem 12 belong to $\mathscr{U}^a_\infty$.
- (E) If the clause C considered initially in the proof of Theorem 13 belongs to $\mathscr{U}^a_\infty$, then all the subsequent clauses in that proof are also duplicate-free.
- (D) In the proof of Theorem 7, the C_{I_m} family of clauses all belong to $\mathscr{U}^a_\infty$.

Thus Table 12 is also a summary of the S-, E- and D-reduction results of $\mathscr{U}^a_\infty$ to $\mathscr{U}^a_2$.

5.5 Summary

We started this section with three goals (**G1**, **G2**, and **G3**). Table 15 summarises the results towards these goals for fragments of metarules relevant to ILP (Table 3). For **G1**, our results are mostly empirical, i.e. the results are the outputs of the reduction algorithms. For **G2**, Table 15 shows that the results are all positive for E-reduction, but mostly negative for S- and D-reduction, especially for Datalog fragments. Similarly, for **G3** the results are again positive for E-reduction but negative for S- and D-reduction for Datalog fragments. We discuss the implications of these results in Section 7.

S-reduction	E-reduction	D-reduction
$P(A) \leftarrow Q(A)$	$P(A) \leftarrow Q(A,B), R(A,B)$	$P(A) \leftarrow Q(A)$
$P(A) \leftarrow Q(A,B), R(A,B)$	$P(A,B) \leftarrow Q(B,A)$	$P(A) \leftarrow Q(A), R(A)$
$P(A,B) \leftarrow Q(A,B)$	$P(A,B) \leftarrow Q(A), R(B)$	$P(A) \leftarrow Q(A,B), R(B)$
$P(A,B) \leftarrow Q(B,A)$		$P(A) \leftarrow Q(A,B), R(A,B)$
$P(A,B) \leftarrow Q(A), R(B)$		$P(A,B) \leftarrow Q(B,A)$
$P(A,B) \leftarrow Q(B), R(A,C), S(A,C)$		$P(A,B) \leftarrow Q(A), R(B)$
$P(A,B) \leftarrow Q(A), R(B,C), S(B,C)$		$P(A,B) \leftarrow Q(A), R(A,B)$
$P(A,B) \leftarrow Q(B,C), R(A,D),$ $S(A,D), T(B,C)$		$P(A,B) \leftarrow Q(A,B), R(A,B)$
		$P(A,B) \leftarrow Q(A,C), R(B,C)$
		$P(A,B) \leftarrow Q(A,C), R(A,D), S(B,C),$ $T(B,D), U(C,D)$
		$P(A,B) \leftarrow Q(B,C), R(A,D), S(B,D)$ $T(C,E), U(E)$
		$P(A,B) \leftarrow Q(B,C), R(A,D), S(B,D),$ $T(C,E), U(C,E)$

Table 13: Reductions of the fragment $\mathcal{U}_5^{\{1,2\}}$

Arities	S-reduction		E-reduction		D-reduction	
a	Bodysize	Cardinality	Bodysize	Cardinality	Bodysize	Cardinality
0	1	1	1	1	2	2
1	1	1	1	1	2	2
2	4	3	5	2	5	10
0,1	1	2	1	2	2	5
0,2	5	4	5	3	5	38
1,2	4	8	2	3	5	12
0,1,2	4	9	2	4	5	16

Table 14: Cardinality and body size of the reductions of $\mathcal{U}_5^a$.

Arities	$\mathcal{C}_\infty^a$			$\mathcal{D}_\infty^a$			$\mathcal{K}_\infty^a$			$\mathcal{U}_\infty^a$		
a	S	E	D	S	E	D	S	E	D	S	E	D
1	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
2	✓	✓	×	✓	✓	×	×	✓	×	×	✓	×
>2	✓	✓	×	×	✓	×	×	✓	×	×	✓	×

Table 15: Existence of a S-, E- or D-reduction of $\mathcal{M}_\infty^a$ to $\mathcal{M}_2^a$. The symbol ✓ denotes that the fragment does have such a reduction. The symbol × denotes that the fragment does not have such a reduction.

6 Experiments

As explained in Section 1, deciding which metarules to use for a given learning task is a major open problem. The problem is the trade-off between efficiency and expressivity: the hypothesis space grows given more metarules (Theorem 1), so we wish to use fewer metarules, but if we use too few metarules then we lose expressivity. In this section we experimentally explore this trade-off. As described in Section 2, Cropper and Muggleton [10] showed that learning with E-reduced sets of metarules can lead to higher predictive accuracies and lower learning times compared to learning with non-E-reduced sets. However, as argued in Section 1, we claim that E-reduction is not always the most suitable form of reduction because it can remove metarules necessary to learn programs with the appropriate specificity. To test this claim, we now conduct experiments that

compare the learning performance of Metagol 2.3.0[19], the main MIL implementation, when given different reduced sets of metarules[20]. We test the null hypothesis:

Null hypothesis 1 There is no difference in the learning performance of Metagol when using different reduced sets of metarules

To test this null hypothesis, we consider three domains: Michalski trains, string transformations, and game rules.

6.1 Michalski trains

In the Michalski trains problems [35] the task is to induce a program that distinguishes eastbound trains from westbound trains. Figure 1 shows an example target program, where the target concept (`f/1`) is that the train has a long carriage with two wheels and another with three wheels.

```
f(X):-
    has_car(X,C1),
    long(C1),
    two_wheels(C1),
    has_car(X,C2),
    long(C2),
    three_wheels(C2).
```

Fig. 1: An example Michalski trains target program. In the Michalski trains domain, a carriage (`car`) can be long or short. A short carriage always has two wheels. A long carriage has either two or three wheels.

6.1.1 Materials

To obtain the experimental data, we first generated 8 random target train programs where the programs are progressively more difficult, where difficulty is measured by the number of literals in the generated program from the easiest task T_1 to the most difficult task T_8. Figure 2 shows the background predicates available to Metagol. We vary the metarules given to Metagol. We use the S-, E-, and D-reductions of the fragment $\mathcal{U}_5^{\{1,2\}}$ (Table 13). In addition, we also consider the $\mathcal{U}_2^{\{1,2\}}$ fragment of the D-reduction of $\mathcal{U}_5^{\{1,2\}}$, i.e. a subset of the D-reduction consisting only of metarules with at most two body literals. This fragment, which we denote as D^*, contains three fewer metarules than the D-reduction of $\mathcal{U}_5^{\{1,2\}}$. Table 16 shows this fragment.

[19] https://github.com/metagol/metagol/releases/tag/2.3.0

[20] Experimental data is available at http://github.com/andrewcropper/mlj19-reduce

has_car/2	has_load/2
short/1	long/1
two_wheels/1	three_wheels/1
roof_open/1	roof_closed/1
zero_load/1	one_load/1
two_load/1	circle/1
triangle/1	rectangle/1

Fig. 2: Background relations available in the trains experiment.

$P(A) \leftarrow Q(A)$	$P(A,B) \leftarrow Q(B,A)$
$P(A) \leftarrow Q(A), R(A)$	$P(A,B) \leftarrow Q(A), R(B)$
$P(A) \leftarrow Q(A,B), R(B)$	$P(A,B) \leftarrow Q(A), R(A,B)$
$P(A) \leftarrow Q(A,B), R(A,B)$	$P(A,B) \leftarrow Q(A,B), R(A,B)$
	$P(A,B) \leftarrow Q(A,C), R(B,C)$

Table 16: The D* fragment, which is the D-reduction of the fragment $\mathscr{U}_5^{\{1,2\}}$ restricted to the fragment $\mathscr{U}_2^{\{1,2\}}$.

6.1.2 Method

For each train task t_i in $\{T_1, \ldots, T_8\}$:

1. Generate 10 training examples of t_i, half positive and half negative
2. Generate 200 testing examples of t_i, half positive and half negative
3. For each set of metarules m in the S-, E-, D-, and D^*-reductions:
   (a) Learn a program for task t_i using the training examples and metarules m
   (b) Measure the predictive accuracy of the learned program using the testing examples

If a program is not found in 10 minutes then no program is returned and every testing example is deemed to have failed. We measure mean predictive accuracies, mean learning times, and standard errors over 10 repetitions.

Task	**S**	**E**	**D**	**D***
T_1	**100** ± 0	**100** ± 0	**100** ± 0	**100** ± 0
T_2	**100** ± 0	**100** ± 0	**100** ± 0	**100** ± 0
T_3	68 ± 5	62 ± 5	**100** ± 0	**100** ± 0
T_4	75 ± 6	75 ± 6	**100** ± 0	**100** ± 0
T_5	92 ± 4	78 ± 6	78 ± 6	**100** ± 0
T_6	52 ± 2	50 ± 0	70 ± 6	**100** ± 0
T_7	95 ± 3	65 ± 5	82 ± 5	**100** ± 0
T_8	55 ± 3	52 ± 2	72 ± 6	**98** ± 2
mean	80 ± 1	73 ± 2	88 ± 2	**100** ± 0

Table 17: Predictive accuracies when using different reduced sets of metarules on the Michalski trains problems.

Task	S	E	D	D*
T1	**0** ± 0	**0** ± 0	**0** ± 0	**0** ± 0
T2	**0** ± 0	**0** ± 0	**0** ± 0	**0** ± 0
T3	424 ± 59	461 ± 56	**0** ± 0	**0** ± 0
T4	322 ± 64	340 ± 61	**0** ± 0	**0** ± 0
T5	226 ± 48	320 ± 59	361 ± 59	**5** ± 2
T6	583 ± 17	600 ± 0	429 ± 51	**7** ± 2
T7	226 ± 44	446 ± 55	243 ± 61	**6** ± 1
T8	550 ± 35	570 ± 30	361 ± 64	**183** ± 40
mean	292 ± 16	342 ± 17	174 ± 16	**25** ± 5

Table 18: Learning times in seconds when using different reduced sets of metarules on the Michalski trains problems. Note that the values are rounded, so 0 represents that a solution was found in under half a second.

```
f(A):-has_car(A,B),f1(A,B).
f1(A,B):-three_wheels(B),has_car(A,C),f2(A,C).
f2(A,B):-roof_open(B),has_car(A,C),has_car(A,C).
```
S program

```
f(A):-has_car(A,B),f1(A,B).
f1(A,B):-f2(A),three_wheels(B).
f2(A):-has_car(A,B),has_car(A,B).
```
E program

```
f(A):-f1(A),f2(A).
f1(A):-has_car(A,B),roof_open(B).
f2(A):-has_car(A,B),three_wheels(B).
```
D program

```
f(A):-f1(A),f2(A).
f1(A):-has_car(A,B),three_wheels(B).
f2(A):-has_car(A,B),f3(B).
f3(A):-long(A),two_wheels(A).
```
D* program

Fig. 3: Example programs learned by Metagol when varying the metarule set. The target program is shown in Figure 1.

6.1.3 Results

Table 17 shows the predictive accuracies when learning with the different sets of metarules. The *D* set generally outperforms the *S* and *E* sets with a higher mean accuracy of 88% vs 80% and 73% respectively. Moreover, the D^* set easily outperforms them all with a

mean accuracy of 100%. A McNemar's test[21] on the D and D^* accuracies confirmed the significance at the $p < 0.01$ level.

Table 18 shows the corresponding learning times when using different reduces sets of metarules. The D set outperforms (has lower mean learning time) the S and E sets, and again the D^* set outperforms them all. A paired t-test[22] on the D and D^* learning times confirmed the significance at the $p < 0.01$ level.

The D^* set performs particularly well on the more difficult tasks. The poor performance of the S and E sets on the more difficult tasks is for one of two reasons. The first reason is that the S- and E-reduction algorithms have removed the metarules necessary to express the target concept. This observation strongly corroborates our claim that E-reduction can be too strong because it can remove metarules necessary to specialise a clause. The second reason is that the S- and E-reduction algorithms produce sets of metarules that are still sufficient to express the target theory but doing so requires a much larger and more complex program, measured by the number of clauses needed.

The performance discrepancy between the D and D^* sets of metarules can be explained by comparing the hypothesis spaces searched. For instance, when searching for a program with 3 clauses, Theorem 1 shows that when using the D set of metarules the hypothesis space contains approximately 10^{24} programs. By contrast, when using the D^* set of metarules the hypothesis space contains approximately 10^{14} programs. As explained in Section 3.2, assuming that the target hypothesis is in both hypothesis spaces, the Blumer bound [3] tells us that searching the smaller hypothesis space will result in less error, which helps to explain these empirical results. Of course, there is the potential for the D^* set to perform worse than the D set when the target theory requires the three removed metarules, but we did not observe this situation in this experiment.

Figure 3 shows the target program for T_8 and example programs learned by Metagol using the various reduced sets of metarules. Only the D^* program is success set equivalent[23] to the target program when restricted to the target predicate `f/1`. In all three cases Metagol discovered that if a carriage has three wheels then it is a long carriage, i.e. Metagol discovered that the literal `long(C2)` is redundant in the target program. Indeed, if we unfold the D^* program to remove the invented predicates then the resulting single clause program is one literal shorter than the target program.

Overall, the results from this experiment suggest that we can reject the null hypothesis, both in terms of predictive accuracies and learning times.

6.2 String transformations

In [38] and [14] the authors evaluate Metagol on 17 real-world string transformation tasks using a predefined (hand-crafted) set of metarules. In this experiment, we compare learning with different metarules on an expanded dataset with 250 string transformation tasks.

[21] A statistical test on paired *nominal* data https://en.wikipedia.org/wiki/McNemar%27s_test

[22] A statistical test on paired *ordinal* data http://www.biostathandbook.com/pairedttest.html

[23] The success set of a logic program P is the set of ground atoms $\{A \in hb(P) | P \cup \{\neg A\}$ has a SLD-refutation$\}$, where $hb(P)$ represents the Herband base of the logic program P. The success set restricted to a specific predicate symbol p is the subset of the success set restricted to atoms containing the predicate symbol p.

6.2.1 Materials

Each string transformation task has 10 examples. Each example is an atom of the form $f(x,y)$ where f is the task name and x and y are strings. Figure 4 shows task *p6* where the goal is to learn a program that filters the capital letters from the input. We supply Metagol with dyadic background predicates, such as `tail`, `dropLast`, `reverse`, `filter_letter`, `filter_uppercase`, `dropWhile_not_letter`, `takeWhile_uppercase`. The full details can be found in the code repository. We vary the metarules given to Metagol. We use the S-, E-, and D-reductions of the fragment $\mathscr{U}_5^{\{2\}}$. We again also use the D-reduction of the fragment $\mathscr{U}_5^{\{2\}}$ restricted to the fragment $\mathscr{U}_2^{\{2\}}$, which is again denoted as D^*.

Input	**Output**
Arthur Joe Juan	AJJ
Jose Larry Scott	JLS
Kevin Jason Matthew	KJM
Donald Steven George	DSG
Raymond Frank Timothy	RFT

Fig. 4: Examples of the *p6* string transformation problem input-output pairs.

6.2.2 Method

Our experimental method is:

1. Sample 50 tasks *Ts* from the set $\{p1,\ldots,p250\}$
2. For each $t \in Ts$:
   (a) Sample 5 training examples and use the remaining examples as testing examples
   (b) For each set of metarules m in the S-, E-, D, and D^*-reductions:
      i. Learn a program p for task t using the training examples and metarules m
      ii. Measure the predictive accuracy of p using the testing examples

If a program is not found in 10 minutes then no program is returned and every testing example is deemed to have failed. We measure mean predictive accuracies, mean learning times, and standard errors over 10 repetitions.

6.2.3 Results

Table 19 shows the mean predictive accuracies and learning times when learning with the different sets of metarules. Note that we are not interested in the absolute predictive accuracy, which is limited by factors such as the low timeout and insufficiency of the BK. We are instead interested in the relative accuracies. Table 19 shows that the D set outperforms the S and E sets, with a higher mean accuracy of 33%, vs 22% and 22% respectively. The D^* set outperforms them all with a mean accuracy of 56%. A McNemar's test on the D and D^* accuracies confirmed the significance at the $p < 0.01$ level.

Table 19 shows the corresponding learning times when varying the metarules. Again, the D set outperforms the S and E sets, and again the D^* set outperforms them all. A

paired t-test on the D and D^* learning times confirmed the significance at the $p < 0.01$ level.

Overall, the results from this experiment give further evidence to reject the null hypothesis, both in terms of predictive accuracies and learning times.

	S	**E**	**D**	**D***
Mean predictive accuracy (%)	22 ± 0	22 ± 0	32 ± 0	**56 ± 1**
Mean learning time (seconds)	467 ± 1	467 ± 1	407 ± 3	**270 ± 3**

Table 19: Experimental results on the string transformation problems.

6.3 Inducing game rules

The general game playing (GGP) framework [25] is a system for evaluating an agent's general intelligence across a wide range of tasks. In the GGP competition, agents are tested on games they have never seen before. In each round, the agents are given the rules of a new game. The rules are described symbolically as a logic program. The agents are given a few seconds to think, to process the rules of the game, and to then start playing, thus producing game traces. The winner of the competition is the agent who gets the best total score over all the games. In this experiment, we use the IGGP dataset [9] which inverts the GGP task: an ILP system is given game traces and the task is to learn a set of rules (a logic program) that could have produced these traces.

6.3.1 Materials

The IGGP dataset contains problems drawn from 50 games. We focus on the eight games shown in Figure 5 which contain BK compatible with the metarule fragments we consider (i.e. the BK contains predicates in the fragment $\mathcal{M}^2_m$). The other games contain predicates with arity greater than two. Each game has four target predicates `legal`, `next`, `goal`, and `terminal`, where the arities depend on the game. Figure 6 shows the target solution for the `next` predicate for the *minimal decay* game. Each game contains training/validate/test data, composed of sets of ground atoms, in a 4:1:1 split. We vary the metarules given to Metagol. We use the S-, E-, and D-reductions of the fragment $\mathcal{D}^{\{1,2\}}_5$. We again also use the D-reduction of the fragment $\mathcal{D}^{\{1,2\}}_5$ restricted to the fragment $\mathcal{D}^{\{1,2\}}_2$, which is again denoted as D^*.

GT attrition	GT chicken
GT prisoner	Minimal decay
Minimal even	Multiple buttons and lights
Scissors paper stone	Untwisty corridor

Fig. 5: IGGP games used in the experiments.

```
next_value(X):-
    true_value(Y),
    succ(X,Y),
    does_player(noop).
next_value(5):-
    does_player(pressButton).
```

Fig. 6: Target solution for the `next` predicate for the *minimal decay* game.

6.3.2 Method

The majority of game examples are negative. We therefore use *balanced accuracy* to evaluate the approaches. Given background knowledge B, sets of positive E^+ and negative E^- testing examples, and a logic program H, we define the number of positive examples as $p = |E^+|$, the number of negative examples as $n = |E^-|$, the number of true positives as $tp = |\{e \in E^+ | B \cup H \models e\}|$, the number of true negatives as $tn = |\{e \in E^- | B \cup H \not\models e\}|$, and the balanced accuracy $ba = (tp/p + tn/n)/2$.

Our experimental method is as follows. For each game g, each task g_t, and each set of metarules m in the S-, E-, D-, and D^*-reductions:

1. Learn a program p using all the training examples for g_t using the metarules m with a timeout of 10 minutes
2. Measure the balanced accuracy of p using the testing examples

If no program is found in 10 minutes then no program is returned and every testing example is deemed to have failed.

6.3.3 Results

Table 20 shows the balanced accuracies when learning with the different sets of metarules. Again, we are not interested in the absolute accuracies only the relative differences when learning using different sets of metarules. The D set outperforms the S and E sets with a higher mean accuracy of 72%, vs 66% and 66% respectively. The D^* set again outperforms them all with a mean accuracy of 73%. A McNemar's test on the D and D^* accuracies confirmed the significance at the $p < 0.01$ level. Table 20 shows the corresponding learning times when varying the metarules. Again, the D set outperforms the S and E sets, and again the D^* set outperforms them all. However, a paired t-test on the D and D^* learning times confirmed the significance only at the $p < 0.08$ level, so the difference in learning times is insignificant. Overall, the results from this experiment suggest that we can reject the null hypothesis in terms of predictive accuracies but not learning times.

	S	**E**	**D**	**D***
Balanced accuracy (%)	66	66	72	**73**
Learning time (seconds)	316	316	327	**296**

Table 20: Experimental results on the IGGP data.

7 Conclusions and further work

As stated in Section 1, despite the widespread use of metarules, there is little work determining which metarules to use for a given learning task. Instead, suitable metarules are assumed to be given as part of the background knowledge, or are used without any theoretical justification. Deciding which metarules to use for a given learning task is a major open challenge [8,10] and is a trade-off between efficiency and expressivity: the hypothesis space grows given more metarules [10,38], so we wish to use fewer metarules, but if we use too few metarules then we lose expressivity. To address this issue, Cropper and Muggleton [10] used E-reduction on sets of metarules and showed that learning with E-reduced sets of metarules can lead to higher predictive accuracies and lower learning times compared to learning with non-E-reduced sets. However, as we claimed in Section 1, E-reduction is not always the most appropriate form of reduction because it can remove metarules necessary to learn programs with the necessary specificity.

To support our claim, we have compared three forms of logical reduction: S-, E-, and D-reduction, where the latter is a new form of reduction based on SLD-derivations. We have used the reduction algorithms to reduce finite sets of metarules. Table 15 summarises the results. We have shown that many sets of metarules relevant to ILP do not have finite reductions (Theorem 7). These negative results have direct (negative) implications for MIL. Specifically, our results mean that, in certain cases, a MIL system, such as Metagol or HEXMIL [32], cannot be given a finite set of metarules from which it can learn any program, such as when learning arbitrary Datalog programs. The results will also likely have implications for other forms of ILP which rely on metarules.

Our experiments compared learning the performance of Metagol when using the different reduced sets of metarules. In general, using the D-reduced set outperforms both the S- and E-reduced sets in terms of predictive accuracy and learning time. Our experimental results give strong evidence to our claim. We also compared a D^*-reduced set, a subset of the D-reduced metarules, which, although derivationally incomplete, outperforms the other two sets in terms of predictive accuracies and learning times.

7.1 Limitations and future work

Theorem 7 shows that certain fragments of metarules do not have finite D-reductions. However, our experimental results show that using D-reduced sets of metarules leads to higher predictive accuracies and lower learning times compared to the other forms of reduction. Therefore, our work now opens up a new challenge of overcoming this negative theoretical result. One idea is to explore whether special metarules, such as a currying metarule [12], could alleviate the issue.

In future work we would also like reduce more general fragments of logic, such as triadic logics, which would allow us to tackle a wider variety or problems, such as more of the games in the IGGP dataset.

We have compared the learning performance of Metagol when using different reduced sets of metarules. However, we have not investigated whether these reductions are optimal. For instance, when considering derivation reductions, it may, in some cases, be beneficial to re-add redundant metarules to the reduced sets to avoid having to derive them through SLD-resolution. In future work, we would like to investigate identifying an optimal set of metarules for a given learning task, or preferably learning which metarules to use for a given learning task.

We have shown that although incomplete the D^*-reduced set of metarules outperforms the other reductions. In future work we would like to explore other methods which sacrifice completeness for efficiency.

We have used the logical reduction techniques to remove redundant metarules. It may also be beneficial to simultaneously reduce metarules and standard background knowledge. The idea of purposely removing background predicates is similar to dimensionality reduction, widely used in other forms of machine learning [59], but which has been under researched in ILP [23]. Initial experiments indicate that this is possible [8, 10], and we aim to develop this idea in future work.

Acknowledgements The authors thank Stephen Muggleton and Katsumi Inoue for discussions on this topic. We especially thank Rolf Morel for valuable feedback on the paper.

References

1. Aws Albarghouthi, Paraschos Koutris, Mayur Naik, and Calvin Smith. Constraint-based synthesis of Datalog programs. In J. Christopher Beck, editor, *Principles and Practice of Constraint Programming - 23rd International Conference, CP 2017, Melbourne, VIC, Australia, August 28 - September 1, 2017, Proceedings*, volume 10416 of *Lecture Notes in Computer Science*, pages 689–706. Springer, 2017.
2. Meghyn Bienvenu. Prime implicates and prime implicants in modal logic. In *Proceedings of the Twenty-Second AAAI Conference on Artificial Intelligence, July 22-26, 2007, Vancouver, British Columbia, Canada*, pages 379–384. AAAI Press, 2007.
3. Anselm Blumer, Andrzej Ehrenfeucht, David Haussler, and Manfred K. Warmuth. Occam's razor. *Inf. Process. Lett.*, 24(6):377–380, 1987.
4. Aaron R. Bradley and Zohar Manna. *The calculus of computation - decision procedures with applications to verification*. Springer, 2007.
5. A. Campero, A. Pareja, T. Klinger, J. Tenenbaum, and S. Riedel. Logical Rule Induction and Theory Learning Using Neural Theorem Proving. *ArXiv e-prints*, September 2018.
6. Alonzo Church. A note on the Entscheidungsproblem. *J. Symb. Log.*, 1(1):40–41, 1936.
7. William W. Cohen. Grammatically biased learning: Learning logic programs using an explicit antecedent description language. *Artif. Intell.*, 68(2):303–366, 1994.
8. Andrew Cropper. *Efficiently learning efficient programs*. PhD thesis, Imperial College London, UK, 2017.
9. Andrew Cropper, Richard Evans, and Mark Law. Inductive general game playing. *arXiv e-prints*, page arXiv:1906.09627, Jun 2019.
10. Andrew Cropper and Stephen H. Muggleton. Logical minimisation of meta-rules within meta-interpretive learning. In Jesse Davis and Jan Ramon, editors, *Inductive Logic Programming - 24th International Conference, ILP 2014, Nancy, France, September 14-16, 2014, Revised Selected Papers*, volume 9046 of *Lecture Notes in Computer Science*, pages 62–75. Springer, 2014.
11. Andrew Cropper and Stephen H. Muggleton. Learning efficient logical robot strategies involving composable objects. In Qiang Yang and Michael Wooldridge, editors, *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015*, pages 3423–3429. AAAI Press, 2015.
12. Andrew Cropper and Stephen H. Muggleton. Learning higher-order logic programs through abstraction and invention. In Subbarao Kambhampati, editor, *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016, New York, NY, USA, 9-15 July 2016*, pages 1418–1424. IJCAI/AAAI Press, 2016.
13. Andrew Cropper and Stephen H. Muggleton. Metagol system. https://github.com/metagol/metagol, 2016.
14. Andrew Cropper and Stephen H. Muggleton. Learning efficient logic programs. *Machine Learning*, 108(7):1063–1083, 2019.
15. Andrew Cropper, Alireza Tamaddoni-Nezhad, and Stephen H. Muggleton. Meta-interpretive learning of data transformation programs. In Katsumi Inoue, Hayato Ohwada, and Akihiro Yamamoto, editors, *Inductive Logic Programming - 25th International Conference, ILP 2015, Kyoto, Japan, August 20-22, 2015, Revised Selected Papers*, volume 9575 of *Lecture Notes in Computer Science*, pages 46–59. Springer, 2015.

16. Andrew Cropper and Sophie Tourret. Derivation reduction of metarules in meta-interpretive learning. In Fabrizio Riguzzi, Elena Bellodi, and Riccardo Zese, editors, *Inductive Logic Programming - 28th International Conference, ILP 2018, Ferrara, Italy, September 2-4, 2018, Proceedings*, volume 11105 of *Lecture Notes in Computer Science*, pages 1–21. Springer, 2018.
17. Evgeny Dantsin, Thomas Eiter, Georg Gottlob, and Andrei Voronkov. Complexity and expressive power of logic programming. *ACM Comput. Surv.*, 33(3):374–425, 2001.
18. Mnacho Echenim, Nicolas Peltier, and Sophie Tourret. Quantifier-free equational logic and prime implicate generation. In Amy P. Felty and Aart Middeldorp, editors, *Automated Deduction - CADE-25 - 25th International Conference on Automated Deduction, Berlin, Germany, August 1-7, 2015, Proceedings*, volume 9195 of *Lecture Notes in Computer Science*, pages 311–325. Springer, 2015.
19. Werner Emde, Christopher Habel, and Claus-Rainer Rollinger. The discovery of the equator or concept driven learning. In Alan Bundy, editor, *Proceedings of the 8th International Joint Conference on Artificial Intelligence. Karlsruhe, FRG, August 1983*, pages 455–458. William Kaufmann, 1983.
20. Richard Evans and Edward Grefenstette. Learning explanatory rules from noisy data. *J. Artif. Intell. Res.*, 61:1–64, 2018.
21. Pierre Flener. Inductive logic program synthesis with DIALOGS. In Stephen Muggleton, editor, *Inductive Logic Programming, 6th International Workshop, ILP-96, Stockholm, Sweden, August 26-28, 1996, Selected Papers*, volume 1314 of *Lecture Notes in Computer Science*, pages 175–198. Springer, 1996.
22. Nuno A. Fonseca, Vítor Santos Costa, Fernando M. A. Silva, and Rui Camacho. On avoiding redundancy in inductive logic programming. In Rui Camacho, Ross D. King, and Ashwin Srinivasan, editors, *Inductive Logic Programming, 14th International Conference, ILP 2004, Porto, Portugal, September 6-8, 2004, Proceedings*, volume 3194 of *Lecture Notes in Computer Science*, pages 132–146. Springer, 2004.
23. Johannes Fürnkranz. Dimensionality reduction in ILP: A call to arms. In *Proceedings of the IJCAI-97 Workshop on Frontiers of Inductive Logic Programming*, pages 81–86, 1997.
24. M. R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979.
25. Michael R. Genesereth, Nathaniel Love, and Barney Pell. General game playing: Overview of the AAAI competition. *AI Magazine*, 26(2):62–72, 2005.
26. Georg Gottlob and Christian G. Fermüller. Removing redundancy from a clause. *Artif. Intell.*, 61(2):263–289, 1993.
27. Georg Gottlob, Nicola Leone, and Francesco Scarcello. On the complexity of some inductive logic programming problems. In Nada Lavrac and Saso Dzeroski, editors, *Inductive Logic Programming, 7th International Workshop, ILP-97, Prague, Czech Republic, September 17-20, 1997, Proceedings*, volume 1297 of *Lecture Notes in Computer Science*, pages 17–32. Springer, 1997.
28. Edith Hemaspaandra and Henning Schnoor. Minimization for generalized boolean formulas. In Toby Walsh, editor, *IJCAI 2011, Proceedings of the 22nd International Joint Conference on Artificial Intelligence, Barcelona, Catalonia, Spain, July 16-22, 2011*, pages 566–571. IJCAI/AAAI, 2011.
29. Marijn Heule, Matti Järvisalo, Florian Lonsing, Martina Seidl, and Armin Biere. Clause elimination for SAT and QSAT. *J. Artif. Intell. Res.*, 53:127–168, 2015.
30. Thomas Hillenbrand, Ruzica Piskac, Uwe Waldmann, and Christoph Weidenbach. From search to computation: Redundancy criteria and simplification at work. In Andrei Voronkov and Christoph Weidenbach, editors, *Programming Logics - Essays in Memory of Harald Ganzinger*, volume 7797 of *Lecture Notes in Computer Science*, pages 169–193. Springer, 2013.
31. William H. Joyner Jr. Resolution strategies as decision procedures. *J. ACM*, 23(3):398–417, 1976.
32. Tobias Kaminski, Thomas Eiter, and Katsumi Inoue. Exploiting answer set programming with external sources for meta-interpretive learning. *TPLP*, 18(3-4):571–588, 2018.
33. Jörg-Uwe Kietz and Stefan Wrobel. Controlling the complexity of learning in logic through syntactic and task-oriented models. In *Inductive logic programming*. Citeseer, 1992.
34. Robert A. Kowalski. Predicate logic as programming language. In *IFIP Congress*, pages 569–574, 1974.
35. J. Larson and Ryszard S. Michalski. Inductive inference of VL decision rules. *SIGART Newsletter*, 63:38–44, 1977.
36. Paolo Liberatore. Redundancy in logic I: CNF propositional formulae. *Artif. Intell.*, 163(2):203–232, 2005.
37. Paolo Liberatore. Redundancy in logic II: 2CNF and Horn propositional formulae. *Artif. Intell.*, 172(2-3):265–299, 2008.
38. Dianhuan Lin, Eyal Dechter, Kevin Ellis, Joshua B. Tenenbaum, and Stephen Muggleton. Bias reformulation for one-shot function induction. In *ECAI 2014 - 21st European Conference on Artificial Intelligence, 18-22 August 2014, Prague, Czech Republic - Including Prestigious Applications of Intelligent Systems (PAIS 2014)*, pages 525–530, 2014.

39. John W. Lloyd. *Foundations of Logic Programming, 2nd Edition*. Springer, 1987.
40. J.W. Lloyd. *Logic for Learning*. Springer, Berlin, 2003.
41. Jerzy Marcinkowski and Leszek Pacholski. Undecidability of the Horn-clause implication problem. In *33rd Annual Symposium on Foundations of Computer Science, Pittsburgh, Pennsylvania, USA, 24-27 October 1992*, pages 354–362, 1992.
42. Pierre Marquis. Consequence finding algorithms. In *Handbook of Defeasible Reasoning and Uncertainty Management Systems*, pages 41–145. Springer, 2000.
43. John McCarthy. Making robots conscious of their mental states. In *Machine Intelligence 15, Intelligent Agents [St. Catherine's College, Oxford, July 1995]*, pages 3–17, 1995.
44. Rolf Morel, Andrew Cropper, and C.-H. Luke Ong. Typed meta-interpretive learning of logic programs. In Francesco Calimeri, Nicola Leone, and Marco Manna, editors, *Logics in Artificial Intelligence - 16th European Conference, JELIA 2019, Rende, Italy, May 7-11, 2019, Proceedings*, volume 11468 of *Lecture Notes in Computer Science*, pages 198–213. Springer, 2019.
45. Stephen Muggleton. Inverse entailment and Progol. *New Generation Comput.*, 13(3&4):245–286, 1995.
46. Stephen Muggleton and Cao Feng. Efficient induction of logic programs. In *Algorithmic Learning Theory, First International Workshop, ALT '90, Tokyo, Japan, October 8-10, 1990, Proceedings*, pages 368–381, 1990.
47. Stephen Muggleton, Luc De Raedt, David Poole, Ivan Bratko, Peter A. Flach, Katsumi Inoue, and Ashwin Srinivasan. ILP turns 20 - biography and future challenges. *Machine Learning*, 86(1):3–23, 2012.
48. Stephen H. Muggleton, Dianhuan Lin, Niels Pahlavi, and Alireza Tamaddoni-Nezhad. Meta-interpretive learning: application to grammatical inference. *Machine Learning*, 94(1):25–49, 2014.
49. Stephen H. Muggleton, Dianhuan Lin, and Alireza Tamaddoni-Nezhad. Meta-interpretive learning of higher-order dyadic Datalog: predicate invention revisited. *Machine Learning*, 100(1):49–73, 2015.
50. Claire Nédellec, Céline Rouveirol, Hilde Adé, Francesco Bergadano, and Birgit Tausend. Declarative bias in ILP. *Advances in inductive logic programming*, 32:82–103, 1996.
51. Shan-Hwei Nienhuys-Cheng and Ronald de Wolf. *Foundations of Inductive Logic Programming*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1997.
52. G.D. Plotkin. *Automatic Methods of Inductive Inference*. PhD thesis, Edinburgh University, August 1971.
53. Luc De Raedt. Declarative modeling for machine learning and data mining. In *Algorithmic Learning Theory - 23rd International Conference, ALT 2012, Lyon, France, October 29-31, 2012. Proceedings*, page 12, 2012.
54. Luc De Raedt and Maurice Bruynooghe. Interactive concept-learning and constructive induction by analogy. *Machine Learning*, 8:107–150, 1992.
55. John Alan Robinson. A machine-oriented logic based on the resolution principle. *J. ACM*, 12(1):23–41, 1965.
56. Manfred Schmidt-Schauß. Implication of clauses is undecidable. *Theor. Comput. Sci.*, 59:287–296, 1988.
57. E.Y. Shapiro. *Algorithmic program debugging*. MIT Press, 1983.
58. Xujie Si, Woosuk Lee, Richard Zhang, Aws Albarghouthi, Paraschos Koutris, and Mayur Naik. Syntax-guided synthesis of Datalog programs. In Gary T. Leavens, Alessandro Garcia, and Corina S. Pasareanu, editors, *Proceedings of the 2018 ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, ESEC/SIGSOFT FSE 2018, Lake Buena Vista, FL, USA, November 04-09, 2018*, pages 515–527. ACM, 2018.
59. David Skillicorn. *Understanding complex datasets: data mining with matrix decompositions*. Chapman and Hall/CRC, 2007.
60. Sten-Åke Tärnlund. Horn clause computability. *BIT*, 17(2):215–226, 1977.
61. Sophie Tourret and Andrew Cropper. SLD-resolution reduction of second-order Horn fragments. In Francesco Calimeri, Nicola Leone, and Marco Manna, editors, *Logics in Artificial Intelligence - 16th European Conference, JELIA 2019, Rende, Italy, May 7-11, 2019, Proceedings*, volume 11468 of *Lecture Notes in Computer Science*, pages 259–276. Springer, 2019.
62. William Yang Wang, Kathryn Mazaitis, and William W. Cohen. Structure learning via parameter learning. In Jianzhong Li, Xiaoyang Sean Wang, Minos N. Garofalakis, Ian Soboroff, Torsten Suel, and Min Wang, editors, *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management, CIKM 2014, Shanghai, China, November 3-7, 2014*, pages 1199–1208. ACM, 2014.
63. Christoph Weidenbach and Patrick Wischnewski. Subterm contextual rewriting. *AI Commun.*, 23(2-3):97–109, 2010.

A Detailed Reduction Results

A.1 Connected ($\mathscr{C}_m^a$) reductions

S-reduction	E-reduction	D-reduction
$P(A,B) \leftarrow Q(A,C)$	$P(A,B) \leftarrow Q(B,C)$	$P(A,A) \leftarrow Q(B,A)$
$P(A,B) \leftarrow Q(B,C)$		$P(A,B) \leftarrow Q(B,A)$
$P(A,B) \leftarrow Q(C,A)$		$P(A,B) \leftarrow Q(B,B)$
$P(A,B) \leftarrow Q(C,B)$		$P(A,B) \leftarrow Q(A,B),R(A,B)$
		$P(A,B) \leftarrow Q(A,C),R(B,C)$
		$P(A,B) \leftarrow Q(A,C),R(A,D),S(B,C),T(B,D),U(C,D)$

Table 21: Reductions of the connected fragment $\mathscr{C}_5^{\{2\}}$

S-reduction	E-reduction	D-reduction
$P(A) \leftarrow Q(A)$	$P(A) \leftarrow Q(B,A)$	$P(A) \leftarrow Q(B,A)$
$P(A) \leftarrow Q(A,B)$	$P(A,B) \leftarrow Q(A)$	$P(A,A) \leftarrow Q(B,A)$
$P(A) \leftarrow Q(B,A)$		$P(A,B) \leftarrow Q(B)$
$P(A,B) \leftarrow Q(A)$		$P(A,B) \leftarrow Q(B,A)$
$P(A,B) \leftarrow Q(B)$		$P(A,B) \leftarrow Q(B,B)$
$P(A,B) \leftarrow Q(A,C)$		$P(A,B) \leftarrow Q(A,B),R(A,B)$
$P(A,B) \leftarrow Q(B,C)$		$P(A,B) \leftarrow Q(A,C),R(B,C)$
$P(A,B) \leftarrow Q(C,A)$		$P(A,B) \leftarrow Q(A,C),R(A,D),S(B,C),T(B,D),U(C,D)$
$P(A,B) \leftarrow Q(C,B)$		

Table 22: Reductions of the connected fragment $\mathscr{C}_5^{\{1,2\}}$

A.2 Datalog ($\mathscr{D}_m^a$) reductions

S-reduction	E-reduction	D-reduction
$P \leftarrow Q$	$P \leftarrow Q$	$P \leftarrow Q$
$P(A) \leftarrow Q(A)$	$P(A) \leftarrow Q(A,B)$	$P \leftarrow Q,R$
$P(A) \leftarrow Q(A,B)$	$P(A,B) \leftarrow Q(B,A)$	$P(A) \leftarrow Q(B,A)$
$P(A) \leftarrow Q(B,A)$	$P(A,B) \leftarrow Q(A),R(B)$	$P(A,A) \leftarrow Q(A)$
$P(A,A) \leftarrow Q(B,A)$		$P(A,A) \leftarrow Q(A,A)$
$P(A,B) \leftarrow Q(A,B)$		$P(A,B) \leftarrow Q(B,A)$
$P(A,B) \leftarrow Q(B,A)$		$P(A,B) \leftarrow Q,R(A,B)$
$P(A,B) \leftarrow Q(A),R(B)$		$P(A,B) \leftarrow Q(A,B),R(A,B)$
$P(A,B) \leftarrow Q(A),R(B,C)$		$P(A,B) \leftarrow Q(A,C),R(B,C)$
$P(A,B) \leftarrow Q(B),R(A,C)$		$P(A,B) \leftarrow Q(B,C),R(A,D)$
$P(A,B) \leftarrow Q(B,C),R(A,D)$		$P(A,B) \leftarrow Q(B,C),R(A,D),S(B,D),T(C,E)$
		$P(A,B) \leftarrow Q(A,C),R(A,D),S(B,C),T(B,D),U(C,D)$
		$P(A,B) \leftarrow Q(B,C),R(A,D),S(C,E),T(B,F),U(D,F)$
		$P(A,B) \leftarrow Q(B,C),R(B,D),S(C,E),T(A,F),U(D,F)$

Table 23: Reductions of the Datalog fragment $\mathscr{D}_5^{\{0,1,2\}}$

S-reduction	E-reduction	D-reduction
$P(A) \leftarrow Q(A)$	$P(A) \leftarrow Q(A,B)$	$P(A) \leftarrow Q(B,A)$
$P(A) \leftarrow Q(A,B)$	$P(A,B) \leftarrow Q(B,A)$	$P(A,A) \leftarrow Q(A)$
$P(A) \leftarrow Q(B,A)$	$P(A,B) \leftarrow Q(A),R(B)$	$P(A,A) \leftarrow Q(A,A)$
$P(A,A) \leftarrow Q(B,A)$		$P(A,B) \leftarrow Q(B,A)$
$P(A,B) \leftarrow Q(A),R(B)$		$P(A,B) \leftarrow Q(A,B),R(A,B)$
$P(A,B) \leftarrow Q(A),R(B,C)$		$P(A,B) \leftarrow Q(A,C),R(B,C)$
$P(A,B) \leftarrow Q(A,B)$		$P(A,B) \leftarrow Q(B,C),R(A,D)$
$P(A,B) \leftarrow Q(B),R(A,C)$		$P(A,B) \leftarrow Q(B,C),R(A,D),S(B,D),T(C,E)$
$P(A,B) \leftarrow Q(B,A)$		$P(A,B) \leftarrow Q(A,C),R(A,D),S(B,C),T(B,D),U(C,D)$
$P(A,B) \leftarrow Q(B,C),R(A,D)$		$P(A,B) \leftarrow Q(B,C),R(A,D),S(C,E),T(B,F),U(D,F)$
		$P(A,B) \leftarrow Q(B,C),R(B,D),S(C,E),T(A,F),U(D,F)$

Table 24: Reductions of the Datalog fragment $\mathscr{D}_5^{\{1,2\}}$

S-reduction	E-reduction	D-reduction
$P(A,A) \leftarrow Q(B,A)$	$P(A,A) \leftarrow Q(B,A)$	$P(A,A) \leftarrow Q(A,A)$
$P(A,B) \leftarrow Q(A,B)$	$P(A,B) \leftarrow Q(B,C),R(A,D)$	$P(A,A) \leftarrow Q(B,A)$
$P(A,B) \leftarrow Q(B,A)$		$P(A,B) \leftarrow Q(B,A)$
$P(A,B) \leftarrow Q(B,C),R(A,D)$		$P(A,B) \leftarrow Q(A,B),R(A,B)$
		$P(A,B) \leftarrow Q(A,C),R(B,C)$
		$P(A,B) \leftarrow Q(B,C),R(A,D)$
		$P(A,B) \leftarrow Q(B,C),R(A,D),S(B,D),T(C,E)$
		$P(A,B) \leftarrow Q(A,C),R(A,D),S(B,C),T(B,D),U(C,D)$
		$P(A,B) \leftarrow Q(B,C),R(A,D),S(C,E),T(B,F),U(D,F)$
		$P(A,B) \leftarrow Q(B,C),R(B,D),S(C,E),T(A,F),U(D,F)$

Table 25: Reductions of the Datalog fragment $\mathscr{D}_5^{\{2\}}$

A.3 Singleton-free ($\mathscr{K}_m^a$) results

S-reduction	E-reduction	D-reduction
$P(A,B) \leftarrow Q(A,B)$	$P(A,B) \leftarrow Q(B,A)$	$P(A,A) \leftarrow Q(A,A)$
$P(A,B) \leftarrow Q(B,A)$	$P(A,B) \leftarrow Q(A,A), R(B,B)$	$P(A,B) \leftarrow Q(B,A)$
$P(A,B) \leftarrow Q(B,C), R(A,D), S(A,D), T(B,C)$	$P(A,B) \leftarrow Q(A,C), R(B,C)$	$P(A,A) \leftarrow Q(A,B), R(B,B)$
		$P(A,B) \leftarrow Q(A,A), R(B,B)$
		$P(A,B) \leftarrow Q(A,B), R(A,B)$
		$P(A,B) \leftarrow Q(A,C), R(B,C)$
		$P(A,B) \leftarrow Q(A,C), R(A,D), S(B,C), T(B,D), U(C,D)$

Table 26: Reductions of the singleton-free fragment $\mathscr{K}_5^{\{2\}}$

S-reduction	E-reduction	D-reduction
$P(A) \leftarrow Q(A)$	$P(A) \leftarrow Q(A,A)$	$P(A) \leftarrow Q(A,A)$
$P(A) \leftarrow Q(A,B), R(A,B)$	$P(A,B) \leftarrow Q(B,A)$	$P(A,A) \leftarrow Q(A)$
$P(A,B) \leftarrow Q(A,B)$	$P(A,B) \leftarrow Q(A), R(B)$	$P(A,B) \leftarrow Q(B,A)$
$P(A,B) \leftarrow Q(B,A)$	$P(A,B) \leftarrow Q(A,C), R(B,C)$	$P(A,A) \leftarrow Q(A,B), R(B,B)$
$P(A,B) \leftarrow Q(A), R(B)$		$P(A,B) \leftarrow Q(A,A), R(B,B)$
$P(A,B) \leftarrow Q(A), R(B,C), S(B,C)$		$P(A,B) \leftarrow Q(A,B), R(A,B)$
$P(A,B) \leftarrow Q(B), R(A,C), S(A,C)$		$P(A,B) \leftarrow Q(A,C), R(B,C)$
$P(A,B) \leftarrow Q(B,C), R(A,D), S(A,D), T(B,C)$		$P(A,B) \leftarrow Q(A,C), R(A,D), S(B,C), T(B,D), U(C,D)$

Table 27: Reductions of the singleton-free fragment $\mathscr{K}_5^{\{1,2\}}$

S-reduction	E-reduction	D-reduction
$P \leftarrow Q$	$P \leftarrow Q$	$P \leftarrow Q$
$P(A) \leftarrow Q(A)$	$P(A) \leftarrow Q(A,A)$	$P \leftarrow Q, R$
$P(A) \leftarrow Q(A,B), R(A,B)$	$P(A,B) \leftarrow Q(B,A)$	$P(A) \leftarrow Q(A,A)$
$P(A,B) \leftarrow Q(A,B)$	$P(A,B) \leftarrow Q(A), R(B)$	$P(A,A) \leftarrow Q(A)$
$P(A,B) \leftarrow Q(B,A)$	$P(A,B) \leftarrow Q(A,C), R(B,C)$	$P(A,B) \leftarrow Q(B,A)$
$P(A,B) \leftarrow Q(A), R(B)$		$P(A,A) \leftarrow Q(A,B), R(B,B)$
$P(A,B) \leftarrow Q(A), R(B,C), S(B,C)$		$P(A,B) \leftarrow Q, R(A,B)$
$P(A,B) \leftarrow Q(B), R(A,C), S(A,C)$		$P(A,B) \leftarrow Q(A,A), R(B,B)$
$P(A,B) \leftarrow Q(B,C), R(A,D), S(A,D), T(B,C)$		$P(A,B) \leftarrow Q(A,B), R(A,B)$
		$P(A,B) \leftarrow Q(A,C), R(B,C)$
		$P(A,B) \leftarrow Q(A,C), R(A,D), S(B,C), T(B,D), U(C,D)$

Table 28: Reductions of the singleton-free fragment $\mathscr{K}_5^{\{0,1,2\}}$

A.4 Duplicate-free ($\mathscr{U}_m^a$) results

S-reduction	E-reduction	D-reduction
$P(A,B) \leftarrow Q(A,B)$	$P(A,B) \leftarrow Q(B,A)$	$P(A,B) \leftarrow Q(B,A)$
$P(A,B) \leftarrow Q(B,A)$	$P(A,B) \leftarrow Q(B,C), R(A,D),$	$P(A,B) \leftarrow Q(A,B), R(A,B)$
$P(A,B) \leftarrow Q(B,C), R(A,D), S(A,D), T(B,C)$	$S(A,D), T(B,C)$	$P(A,B) \leftarrow Q(A,C), R(B,C)$
		$P(A,B) \leftarrow Q(A,B), R(A,C), S(A,C)$
		$P(A,B) \leftarrow Q(A,B), R(A,C), S(C,D), T(C,D)$
		$P(A,B) \leftarrow Q(B,C), R(A,D), S(A,D), T(B,C)$
		$P(A,B) \leftarrow Q(B,C), R(A,D), S(B,C), T(B,D)$
		$P(A,B) \leftarrow Q(A,C), R(A,D), S(B,C), T(B,D), U(C,D)$
		$P(A,B) \leftarrow Q(B,C), R(A,D), S(B,D), T(C,E), U(C,E)$
		$P(A,B) \leftarrow Q(B,C), R(C,D), S(A,E), T(B,E), U(C,D)$

Table 29: Reductions of the fragment $\mathscr{U}_5^{\{2\}}$

S-reduction	E-reduction	D-reduction
$P(A) \leftarrow Q(A)$	$P(A) \leftarrow Q(A,B), R(A,B)$	$P(A) \leftarrow Q(A)$
$P(A) \leftarrow Q(A,B), R(A,B)$	$P(A,B) \leftarrow Q(B,A)$	$P(A) \leftarrow Q(A), R(A)$
$P(A,B) \leftarrow Q(A,B)$	$P(A,B) \leftarrow Q(A), R(B)$	$P(A) \leftarrow Q(A,B), R(B)$
$P(A,B) \leftarrow Q(B,A)$		$P(A) \leftarrow Q(A,B), R(A,B)$
$P(A,B) \leftarrow Q(A), R(B)$		$P(A,B) \leftarrow Q(B,A)$
$P(A,B) \leftarrow Q(B), R(A,C), S(A,C)$		$P(A,B) \leftarrow Q(A), R(B)$
$P(A,B) \leftarrow Q(A), R(B,C), S(B,C)$		$P(A,B) \leftarrow Q(A), R(A,B)$
$P(A,B) \leftarrow Q(B,C), R(A,D), S(A,D), T(B,C)$		$P(A,B) \leftarrow Q(A,B), R(A,B)$
		$P(A,B) \leftarrow Q(A,C), R(B,C)$
		$P(A,B) \leftarrow Q(A,C), R(A,D), S(B,C), T(B,D), U(C,D)$
		$P(A,B) \leftarrow Q(B,C), R(A,D), S(B,D), T(C,E), U(E)$
		$P(A,B) \leftarrow Q(B,C), R(A,D), S(B,D), T(C,E), U(C,E)$

Table 30: Reductions of the fragment $\mathscr{U}_5^{\{1,2\}}$

S-reduction	E-reduction	D-reduction
$P \leftarrow Q$	$P \leftarrow Q$	$P \leftarrow Q$
$P(A) \leftarrow Q(A)$	$P(A) \leftarrow Q(A,B), R(A,B)$	$P(A) \leftarrow Q(A)$
$P(A) \leftarrow Q(A,B), R(A,B)$	$P(A,B) \leftarrow Q(B,A)$	$P(A,B) \leftarrow Q(B,A)$
$P(A,B) \leftarrow Q(B,A)$	$P(A,B) \leftarrow Q(A), R(B)$	$P \leftarrow Q, R$
$P(A,B) \leftarrow Q(A,B)$		$P(A) \leftarrow Q, R(A)$
$P(A,B) \leftarrow Q(A), R(B)$		$P(A) \leftarrow Q(A), R(A)$
$P(A,B) \leftarrow Q(A), R(B,C), S(B,C)$		$P(A) \leftarrow Q(A,B), R(B)$
$P(A,B) \leftarrow Q(B), R(A,C), S(A,C)$		$P(A) \leftarrow Q(A,B), R(A,B)$
$P(A,B) \leftarrow Q(B,C), R(A,D), S(A,D), T(B,C)$		$P(A,B) \leftarrow Q, R(A,B)$
		$P(A,B) \leftarrow Q(A), R(B)$
		$P(A,B) \leftarrow Q(A), R(A,B)$
		$P(A,B) \leftarrow Q(A,B), R(A,B)$
		$P(A,B) \leftarrow Q(A,C), R(B,C)$
		$P(A,B) \leftarrow Q(A,C), R(A,D), S(B,C), T(B,D), U(C,D)$
		$P(A,B) \leftarrow Q(B,C), R(A,D), S(B,D), T(C,E), U(E)$
		$P(A,B) \leftarrow Q(B,C), R(A,D), S(B,D), T(C,E), U(C,E)$

Table 31: Reductions of the fragment $\mathscr{U}_5^{\{0,1,2\}}$