# Derivation reduction of metarules in meta-interpretive learning

Andrew Cropper[1] and Sophie Tourret[2]

[1] University of Oxford
andrew.cropper@cs.ox.ac.uk
[2] Max Planck Institute for Informatics
stourret@mpi-inf.mpg.de

**Abstract.** Meta-interpretive learning (MIL) is a form of inductive logic programming. MIL uses second-order Horn clauses, called metarules, as a form of declarative bias. Metarules define the structures of learnable programs and thus the hypothesis space. Deciding which metarules to use is a trade-off between efficiency and expressivity. The hypothesis space increases given more metarules, so we wish to use fewer metarules, but if we use too few metarules then we lose expressivity. A recent paper used Progol's entailment reduction algorithm to identify irreducible, or minimal, sets of metarules. In some cases, as few as two metarules were shown to be sufficient to entail all hypotheses in an infinite language. Moreover, it was shown that compared to non-minimal sets, learning with minimal sets of metarules improves predictive accuracies and lowers learning times. In this paper, we show that entailment reduction can be too strong and can remove metarules necessary to make a hypothesis more specific. We describe a new reduction technique based on derivations. Specifically, we introduce the *derivation reduction* problem, the problem of finding a finite subset of a Horn theory from which the whole theory can be derived using SLD-resolution. We describe a derivation reduction algorithm which we use to reduce sets of metarules. We also theoretically study whether certain sets of metarules can be derivationally reduced to minimal finite subsets. Our experiments compare learning with entailment and derivation reduced sets of metarules. In general, using derivation reduced sets of metarules outperforms using entailment reduced sets of metarules, both in terms of predictive accuracies and learning times.

## 1 Introduction

Meta-interpretive learning (MIL) [4, 6, 25] is a form of inductive logic programming (ILP). MIL uses second-order Horn clauses, called *metarules*, as a form of declarative bias [28]. Metarules define the structure of learnable programs, which in turn defines the hypothesis space. For instance, to use MIL to learn the *grandparent/2* relation given the *parent/2* relation, the *chain* metarule would be suitable:

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

The letters $P$, $Q$, and $R$ denote existentially quantified second-order variables (i.e. variables that can be bound to predicate symbols), and the letters $A$, $B$ and $C$ denote universally quantified first-order variables (i.e. variables that can bound to constant symbols). Given this metarule, the background *parent/2* relation, and examples of the *grandparent/2* relation, MIL uses a Prolog meta-interpreter to generate a proof of the examples by finding substitutions for the second-order variables. In this scenario, MIL could perform the substitutions {P/grandparent, Q/parent, R/parent} to induce the theory:

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

Many ILP systems, such as BLIP [7], Clint [29], DIALOGS [9], MOBAL [14], and MIL-HEX [13], use metarules[3] (or variants of them). Non-ILP program induction systems, such as ProPPR [32], SYNTH [1], and DILP [8], also use variants of metarules. However, despite their widespread use, there is little work determining which metarules to use for a given task. Instead, suitable metarules are typically assumed to be given as part of the background knowledge[4].

In MIL, deciding which metarules to use is a trade-off between efficiency and expressivity. The hypothesis space increases given more metarules (Theorem 3.3), 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 monadic metarules. To address this issue, Cropper and Muggleton [3] used Progol's entailment-reduction algorithm [23] to identify irreducible, or minimal, sets of metarules. Their approach removed entailment redundant clauses from sets of metarules, where a clause $C$ is entailment redundant in a clausal theory $T \cup \{C\}$ when $T \models C$. To illustrate this form of redundancy, consider the clausal theory:

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

The clause $C_2$ is entailment redundant because it is a logical consequences of $C_1$.

Cropper and Muggleton showed that, in some cases, as few as two metarules are sufficient to entail an infinite fragment of second-order dyadic datalog. Moreover, they showed that learning with minimal sets of metarules improves predictive accuracies and reduces learning times compared to non-minimal sets. However, entailment reduction is not always the most appropriate form of reduction. To illustrate this point, suppose you want to learn the *father/2* relation given the background relations *parent/2*, *male/1*, and *female/1*. Then a suitable hypothesis would be:

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

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

$$M_1 = P(A,B) \leftarrow Q(A,B)$$
$$M_2 = P(A,B) \leftarrow Q(A,B), R(A)$$

Running entailment reduction on these metarules would remove $M_2$ because it is a logical consequence of $M_1$. But it is impossible to learn the intended *father/2* theory 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 describe a new form of reduction based on derivations. Let $\vdash$ represent derivability in SLD-resolution [15], 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 reduction and derivation reduction, consider the metarules:

[3] Metarules are also called *program schemata* [9], *second-order schemata* [29], and *clause templates* [1]

[4] Assuming suitable background knowledge, especially syntactic bias, is a frequent criticism of ILP from other areas of machine learning

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

Running entailment reduction on these would leave the single metarule $M_1$ because it entails the rest of the theory. By contrast, performing 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.

### 1.1 Contributions

Our main contributions are:

- We introduce the derivation reduction problem, the problem of removing derivationally redundant clauses from a clausal theory, and show that the problem is undecidable in general (Section 3)
- We introduce a derivation reduction algorithm (Section 3)
- We run derivation and entailment reduction on finite sets of metarules to identify minimal sets (Section 4)
- We theoretically study whether sets of metarules can be derivationally reduced (Section 4)
- We experimentally compare learning with derivation and entailment reduced metarules, where the results show that using the former set results in higher predictive accuracies and lower learning times (Section 5)

## 2 Related work

*Meta-interpretive learning* Although the study of metarules has implications for many ILP approaches [1,7–9,14,29,32], we are primarily motivated by MIL. MIL is a form of ILP based on a Prolog meta-interpreter. 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, 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 onto their corresponding metarules, allowing for a form of ILP which supports predicate invention and learning recursive theories.

*Metarules* Metarules were introduced in the Blip system [7]. Kietz and Wrobel [14] 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 resolution. This point is important because the ability to derive more complex metarules through resolution allows us to start from smaller sets of primitive or core metarules. The focus of this paper is identifying such core sets.

Using metarules to build a logic program is similar to the use of refinement operators in ILP [26, 31] to build a definite clause literal-by-literal[5]. As with refinement operators, it seems reasonable to ask about completeness and irredundancy of a set of metarules, which we explore in this paper.

*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 [3, 10]. In general, simplifying or reducing a theory often makes a theory easier to understand and use, and may also have computational efficiency advantages.

Plotkin [27] introduced methods to decide whether a clause is subsumption redundant in a first-order clausal theory. The same problem, and slight variants, has been extensively studied in the propositional case [18, 19]. Removing redundant clauses has numerous applications, such as to improve the efficiency of SAT [12]. In contrast to these works, we focus on reducing theories formed of second-order Horn clauses, which to our knowledge has not yet been extensively explored. Another difference is that we study redundancy based on SLD-derivations. Langlois et al. [16] also considered derivations. They studied combinatorial problems for propositional Horn clauses. By contrast, we focus on derivationally reducing sets of second-order Horn clauses.

The work most relevant to this paper is by Cropper and Muggleton [3]. They used Progol's entailment-reduction algorithm [23] 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 language. However, they only considered small and highly constrained fragments of metarules. In particular, they focused on metarules where each literal is dyadic and each term variable appears exactly twice (we call this fragment *exactly-two-connected*, see Definition 4.2). In this paper, we go beyond entailment reduction and introduce derivation reduction. We also consider more general fragments of metarules.

## 3 Logical reduction

We now introduce the derivation reduction problem, the problem of removing derivationally redundant clauses from a clausal theory. Before introducing this problem, we describe preliminary notation and also describe *entailment reduction*, to which we compare our new approach.

### 3.1 Preliminaries

We assume familiarity with logic programming notation [21], 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. Most of the concepts introduced in this section can be defined for any resolution-based proof system, but, because MIL is based on a Prolog meta-interpreter, we focus on SLD-resolution [15]. To identify clauses derivable from a theory, we first define a function $R^n(T)$ of a Horn theory $T$ as:

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

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

We use this definition to define the SLD-closure of a Horn theory:

**Definition 3.1 (SLD-closure).** *The SLD-closure $R^*(T)$ of a Horn theory $T$ is: $\bigcup_{n \in \mathcal{N}} R^n(T)$*

We can now state our notion of derivability:

**Definition 3.2 (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 also introduce *k-derivability*:

**Definition 3.3 (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 $C \in R^k(T)$.*

Some definitions and results in this section rely on Kowalski's *subsumption theorem* for SLD-resolution [15], which is based on SLD-deductions [15]:

**Definition 3.4 (SLD-deduction).** *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 denote a SLD-deduction restricted by k-derivability as $\vdash_{d_k}$. To illustrate the difference between $\vdash$ and $\vdash_d$, consider the clauses $M_1$ to $M_4$ defined in the introduction. We have $\{M_1\} \vdash_d \{M_2, M_3, M_4\}$ but $\{M_1\} \nvdash \{M_2, M_3, M_4\}$. Kowalski's *subsumption theorem* shows the relationship between SLD-deductions and logical entailment:

**Theorem 3.1 (SLD-subsumption theorem).** *Let $T$ be a Horn theory and $C$ be a Horn Clause. Then $T \models C$ if and only if $T \vdash_d C$.*

A more general version of this theorem also applies to unconstrained resolution [26].

### 3.2 Entailment reduction

Muggleton [23] provided two definitions for eliminating entailment redundant clauses from a clausal theory:

**Definition 3.5 (Entailment redundant clause).** *The clause $C$ is entailment redundant in the clausal theory $T \cup \{C\}$ whenever $T \models C$.*

**Definition 3.6 (Entailment reduced theory).** *A clausal theory is entailment reduced if and only if it does not contain any redundant clauses.*

**Algorithm 1** k-bounded entailment reduction

**Input:** a Horn theory $T$ and a natural number $k$
**Output:** a $k$-bounded E-reduced theory $T'$
  $T' = T$
  **while** there is a clause $C$ in $T'$ such that $T' \setminus \{C\} \vdash_{d_k} C$ **do**
    $T' = T' \setminus \{C\}$
  **end while**

If $C$ is entailment redundant in $T \cup \{C\}$ then $T$ is entailment equivalent to $T \cup \{C\}$ because $T \models T \cup \{C\}$ and $T \cup \{C\} \models T$. Muggleton's definitions apply to clauses, but can easily be adapted to Horn clauses.

Because entailment between arbitrary Horn clauses is undecidable [22], determining whether a Horn clause is entailment redundant in a Horn theory is also undecidable[6]. Algorithm 1 finds a k-bounded entailment reduction (henceforth called an *E-reduction*) $T'$ of a Horn theory $T$.

In Section 4, we use a Prolog implementation of Algorithm 1 to find E-reduced sets of metarules.

### 3.3 Derivation reduction

We now describe a new form of reduction based on derivability. We first define *derivationally redundant* clauses:

**Definition 3.7 (Derivationally redundant clause).** *A Horn clause $C$ is derivationally redundant in the Horn theory $T \cup \{C\}$ if and only if $T \vdash C$.*

We can now define *derivationally reduced* theories:

**Definition 3.8 (Derivationally reduced theory).** *A Horn theory is derivationally reduced if and only if it does not contain any derivationally redundant clauses.*

We now define the *derivation reduction problem*:

**Definition 3.9 (Derivation reduction problem).** *Given a Horn theory $T$, the derivation reduction problem is to find a finite theory $T' \subseteq T$ such that (1) $T' \vdash C$ for every Horn clause $C$ in $T$, and (2) $T'$ is derivationally reduced.*

Note that a solution to the derivation reduction problem must be a finite set. For convenience, we name the output of the derivation reduction problem:

**Definition 3.10 (Derivation reduction).** *Let $T$ and $T'$ be the input and output respectively from a derivation reduction problem. Then we call $T'$ a derivation reduction (or D-reduction) of $T$.*

The following proposition outlines the connection between an E-reduction and a D-reduction:

**Proposition 3.1.** *Let $T$ be a Horn theory, $T_E$ be an E-reduction of $T$, and $T_D$ be a D-reduction of $T$. Then $T_E \subseteq T_D$.*

[6] Entailment reduction is decidable in the case of a function-free theory [23]

Algorithm 2 finds a D-reduction $T'$ of a Horn theory $T$. Note that a fragment 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 you can resolve the first body literal of $C_2$ with $C_1$ to derive $C_3$ (with variable renaming). Another D-reduction of $T$ is $\{C_1,C_3\}$ because you can likewise resolve the first body literal of $C_3$ with $C_1$ to derive $C_2$.

As with the entailment reduction problem, the derivation reduction problem is undecidable for Horn theories:

**Theorem 3.2 (Horn decidability).** *The derivation 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 [11], 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 [30], so the assumption cannot hold. Therefore, the problem must be undecidable.

**Algorithm 2** Derivation reduction

**Input:** a Horn theory $T$
**Output:** a D-reduced theory $T'$
  $T' = T$
  **while** there is a clause $C$ in $T'$ such that $T' \setminus \{C\} \vdash C$ **do**
    $T' = T' \setminus \{C\}$
  **end while**

In future work, described in Section 6, we want to study the decidability of the derivation problem for other forms of logic, such as datalog. To overcome the aforementioned undecidability issue, we use a k-bounded d-reduction algorithm (algorithm omitted for brevity). The k-bounded version is similar to Algorithm 2 but additionally takes as input a resolution depth bound $k$ which is used to constrain the SLD-derivability check step. This k-bounded version has the worst-case time complexity:

**Proposition 3.2 (k-bounded derivation reduction complexity).** *Given a Horn theory $T$ and a natural number $k$, k-bounded derivation reduction requires at most $O(|T|^k)$ resolutions.*

**Sketch proof 1** In the worst case the algorithm searches the whole SLD-tree which has a maximum branching factor $|T|$ and a maximum depth $k$. Thus, the overall complexity is $O(|T|^k)$.

In Section 4, we use the k-bounded entailment and derivation reduction algorithms to logically reduce sets of metarules. From this point onwards, any reference to the entailment or derivation reduction algorithms refer to the k-bounded versions.

### 3.4 Language class and hypothesis space

In Section 4 we logically reduce fragments of second-order datalog formed of metarules, where a fragment of a theory is a syntactically restricted subset of that theory [2]. We make explicit our notion of a metarule, which is a second-order Horn clause:

**Definition 3.11 (Second-order Horn clause).** *A second-order Horn clause is of the form:*

$$A_0 \leftarrow A_1, ..., A_m$$

*where each $A_i$ is a literal of the form $P(T_1,...,T_n)$ 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.*

We denote the language of second-order datalog as $\mathcal{H}$, which we further restrict. Our first restriction is on the syntactic form of clauses in $\mathcal{H}$:

**Definition 3.12 (The fragment $\mathcal{H}^a_m$).** *We denote as $\mathcal{H}^a_m$ the fragment of $\mathcal{H}$ where each literal has arity at most $a$ and each clause has at most $m$ literals in the body.*

Having defined this fragment we can characterise the size of the hypothesis space of a MIL learner given metarules restricted to this fragment. The following result generalises previous results [4, 20]:

**Theorem 3.3 (Number of programs in $\mathcal{H}^a_m$).** *Given $p$ predicate symbols and $k$ metarules (not necessarily distinct), the number of $\mathcal{H}^a_m$ programs expressible with at most $n$ clauses is $O((p^{m+1}k)^n)$.*

*Proof.* The number of clauses which can be constructed from a $\mathcal{H}^a_m$ metarule given $p$ predicate symbols is at most $p^{m+1}$ because for a given metarule there are potentially $m+1$ predicate variables with $p^{m+1}$ possible substitutions. Therefore the set of such clauses $S_{k,p,m}$ which can be formed from $k$ distinct $\mathcal{H}^a_m$ metarules using $p$ predicate symbols has cardinality at most $kp^{m+1}$. It follows that the number of programs which can be formed from a selection of $n$ rules chosen from $S_{k,p,m}$ is at most $O((p^{m+1}k)^n)$.

Theorem 3.3 shows that the MIL hypothesis space increases given more metarules, which suggests that we should remove redundant metarules. The next section explores this idea.

## 4 Reduction of metarules

We now logically reduce fragments of second-order datalog, where the fragments correspond to sets of metarules. The goal is to identify a finite minimal set of metarules from which a larger (possibly infinite) set can be derived. To reason about metarules using Prolog (i.e. when running the Prolog implementations of the reduction algorithms), we use a method called encapsulation [3] which transforms a second-order logic program to a first-order logic program. To aid readability of the results, we present non-encapsulated (i.e second-order) metarules.

We focus on fragments of second-order datalog useful to ILP. We follow standard ILP convention [3, 8, 26] and only consider fragments consisting of connected clauses:

**Definition 4.1 (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.*

We further restrict the fragments using syntactic restrictions on their clauses, namely on literal arity and clause body size (Definition 3.12). We denote the case that each literal has arity of exactly $a$ as $=$. For instance, the fragment where each clause has at most two literals in the body and each literal has arity of exactly 3 is denoted as $\mathcal{H}_2^{3=}$. We consider two fragments: *exactly two-connected* and *two-connected*, both of which contain only connected clauses. Our goal is to first identify k-bounded E-reduction and D-reductions for these fragments using the reduction algorithms described in Section 3. To identify reductions for an infinite fragment, such as $\mathcal{H}_*^a$, we first run the reduction algorithms on the sub-fragment $\mathcal{H}_5^a$ using a resolution bound 10 (i.e. $k=10$). Having found k-bounded reductions for the fragment $\mathcal{H}_5^a$, our goal is to then theoretically determine whether larger (preferably infinite) sets can be derived from these reductions.

### 4.1 Exactly-two-connected fragment

We first consider an *exactly-two-connected* fragment, studied by Cropper and Muggleton [3]. The restriction is:

**Definition 4.2 (Exactly-two-connected clause).** *A clause is exactly-two-connected if each term variable appears exactly twice.*

We denote the exactly-two-connected fragment of $\mathcal{H}$ as $\mathcal{E}$. Figure 1(a) shows the results of applying the entailment and derivation reduction algorithms to $\mathcal{E}_5^{2=}$. Both algorithms return the same reduced set of two metarules in $\mathcal{E}_2^{2=}$. This result corroborates the result of Cropper and Muggleton [3]. Figure 1(b) shows the results of applying the entailment and derivation reduction algorithms to $\mathcal{E}_5^2$, a fragment not studied by Cropper and Muggleton. Again, both algorithms return the same reduced set of metarules in $\mathcal{E}_2^2$. We now show that $\mathcal{E}_\infty^2$ has the same D-reduction as $\mathcal{E}_2^2$:

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

(a) Reductions of $\mathcal{E}_5^{2=}$

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

(b) Reductions of $\mathcal{E}_5^2$

Fig. 1: Figures (a) and (b) show the results of applying the E-reduction and D-reduction algorithms to the corresponding fragments

**Theorem 4.1 ($\mathcal{E}_\infty^2$ reducibility).** *The fragment $\mathcal{E}_\infty^2$ has the same D-reduction as $\mathcal{E}_2^2$.*

*Proof.* See appendix A for the proof.

An immediate consequence of Theorem 4.1 is the result:

**Corollary 4.1.** *The fragment $\mathcal{E}^2_\infty$ has the same E-reduction as $\mathcal{E}^2_2$.*

*Proof.* Follows from Theorem 4.1 and Proposition 3.1.

### 4.2 Two-connected fragment

A common constraint in ILP is to require that all the variables in a clause appear at least twice [24, 29]. We study a slight variant of this constraint which we call *two-connected*:

**Definition 4.3 (Two-connected clause).** *A clause is two-connected if each term variable appears in at least two literals*

We denote the two-connected fragment of $\mathcal{H}$ as $\mathcal{K}$. Figure 2 shows the results of applying the entailment and derivation reduction algorithms to $\mathcal{K}^{2=}_5$. Unlike the exactly two-connected fragment, the algorithms do not return the same reduced sets of metarules. Whereas the E-reduced set is in $\mathcal{K}^{2=}_2$, the D-reduced set is not in $\mathcal{K}^{2=}_2$. In fact, although the majority of clauses have been removed, the D-reduced set still contains a clause with five body literals. Figure 3 shows the results of applying the entailment and derivation reduction algorithms to $\mathcal{K}^2_5$. Again, whereas the E-reduced set is in $\mathcal{K}^2_2$, the D-reduced set is not in $\mathcal{K}^2_2$, and again contains a clause with five body literals. We now show that the D-reduction of $\mathcal{K}^2_5$ is not in $\mathcal{K}^2_2$:

**Proposition 4.1 ($\mathcal{K}^2_5$ irreducibility).** *There is no D-reduction of $\mathcal{K}^2_5$ in $\mathcal{K}^2_2$*

**Sketch proof 2** We use the clause $P_0(x_1,x_2) \leftarrow P_1(x_1,x_3),P_2(x_1,x_4),P_3(x_2,x_3),P_4(x_2,x_4),P_5(x_3,x_4)$ as a counter example. We explore the different ways to derive this clause from strictly smaller clauses. We reach a contradiction each time. See appendix B for the full proof.

We also show that $\mathcal{K}^2_\infty$ has no D-reduction:

**Theorem 4.2 ($\mathcal{K}^2_\infty$ irreducibility).** *$\mathcal{K}^2_\infty$ has no D-reduction*

**Sketch proof 3** We define a transformation that turns an irreducible clause, such as the counter example in Proposition 4.1, into a larger irreducible clause. See appendix B for the full proof.

### 4.3 Discussion

We have used the entailment and derivation reduction algorithms to reduce four fragments of second-order datalog, corresponding to sets of metarules. Theorem 4.2 shows that certain fragments do not have finite reductions. This result has implications for the completeness of any ILP system which relies on metarules. In MIL, for instance, the result implies incompleteness when learning programs in the fragment $\mathcal{K}^2_\infty$.

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

Fig. 2: Reductions of $\mathcal{K}_5^{2=}$

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

Fig. 3: Reductions of $\mathcal{K}_5^2$

## 5 Experiments

As explained in Section 1, entailment reduction can be too strong and can remove metarules necessary to make a hypothesis more specific. The contrast between entailment and derivation reduction was shown in the previous section, where in all cases the E-reductions are a subset of the D-reductions. However, as shown in Theorem 3.3, the MIL hypothesis space increases given more metarules, which suggests that we should use fewer metarules. In this section we experimentally explore this tradeoff between expressivity and efficiency. Specifically, we describe an experiment[7] that compares learning with the different reduced sets of metarules. We test the null hypothesis:

**Null hypothesis 1** There is no difference in learning performance when using E-reduced and D-reduced sets of metarules

*Materials* We use Metagol [5], the main MIL implementation, to compare learning with the E-reduced and D-reduced sets of metarules for the fragment $\mathcal{K}_5^2$. We also compare a third set which we call *D*-reduced*, which is the D-reduced set but without the irreducible metarule with 5 literals in the body. We compare the sets of metarules on the Michalski trains problems [17], where the task is to induce a hypothesis that distinguishes five eastbound trains from five westbound trains. To generate the experimental data, we first randomly generated 8 target train programs, where the programs are progressively more difficult measured by the number of literals.

[7] All code and data used in the experiments are available at https://github.com/andrewcropper/ilp18-dreduce

*Method* Our experimental method is as follows. For each target program:

1. Generate 10 training examples, half positive and half negative
2. Generate 200 testing examples, half positive and half negative
3. For each set of metarules $m$:
   (a) Learn a program $p$ using the training examples and metarules $m$ with a timeout of 10 minutes
   (b) Measure the predictive accuracy of $p$ using the testing examples

We repeat the above method 20 times, and measure mean predictive accuracies, learning times, and standard errors.

*Results* Figure 4(a) shows the predictive accuracies when learning with the different sets of metarules. In 6/8 tasks, the D-reduced set has higher predictive accuracies than the E-reduced set. The D-reduced and D*-reduced sets have similar levels of performance, except on the most difficult task 8. A McNemar's test on the D-reduced and D*-reduced accuracies confirmed the significance at the $p < 0.01$ level. On task 8, the D*-reduced set greatly outperforms the other two sets. The poor performance with the D-reduced set on task 8 is because Metagol often times out when using these metarules, which can be explained by the larger hypothesis space searched (Theorem 3.3). Specifically, when searching for a program with 3 clauses, the D-reduced space contains $4.27^{23}$ programs, whereas the D*-reduced space contains $1.51^{13}$ programs. When searching for a program with 4 clauses, the D-reduced space contains $3.21^{31}$ programs, whereas the D*-reduced space contains $3.72^{17}$ programs.

Figure 4(b) shows the learning times when learning using the different reduced sets of metarules. Again, the D-reduced set has lower learning times compared to the E-reduced set, and again the D*-reduced set outperforms the D-reduced set on the most difficult task 8. A paired t-test on the D-reduced and D*-reduced learning times confirmed the significance at the $p < 0.01$ level. Figure 5 shows the target program for task 8 and example programs learned by Metagol using the various reduced sets of metarules. In both cases, the null hypothesis is refuted, both in terms of predictive accuracies and learning times.

| Task | E-reduction | D-reduction | D*-reduction |
|---|---|---|---|
| T1 | 95 ± 1 | 100 ± 0 | 100 ± 0 |
| T2 | 99 ± 1 | 100 ± 0 | 100 ± 0 |
| T3 | 56 ± 3 | 96 ± 2 | 96 ± 2 |
| T4 | 69 ± 4 | 96 ± 2 | 96 ± 2 |
| T5 | 59 ± 3 | 93 ± 3 | 93 ± 3 |
| T6 | 50 ± 1 | 96 ± 3 | 96 ± 3 |
| T7 | 68 ± 4 | 95 ± 2 | 95 ± 2 |
| T8 | 54 ± 3 | 60 ± 3 | 90 ± 3 |

(a)

| Task | E-reduction | D-reduction | D*-reduction |
|---|---|---|---|
| T1 | 0.01 ± 0 | 0 ± 0 | 0 ± 0 |
| T2 | 0.01 ± 0 | 0 ± 0 | 0 ± 0 |
| T3 | 431 ± 59 | 0.01 ± 0 | 0.01 ± 0 |
| T4 | 300 ± 68 | 0 ± 0 | 0.01 ± 0 |
| T5 | 427 ± 60 | 1 ± 0.3 | 1 ± 0.41 |
| T6 | 600 ± 0 | 1 ± 0.41 | 1 ± 0.42 |
| T7 | 917 ± 535 | 1 ± 0.27 | 1 ± 0.36 |
| T8 | 487 ± 51 | 360 ± 67 | 26 ± 5 |

(b)

Fig. 4: Figures (a) and (b) show the predictive accuracies (%) and learning times (seconds rounded to 2 decimal places) respectively when using different reduced sets of metarules on the Michalski trains problems

Target program

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

E-reduction program

```
f(A):-has_car(A,B),f1(A,B).
f1(A,B):-has_car(A,C),f2(C,B).
f2(A,B):-long(A),three_wheels(B).
```

D-reduction program

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

D*-reduction 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).
```

Fig. 5: Example programs learned by Metagol when varying the metarule set. Only the D*-reduction program is success set equivalent 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. In fact, if you unfold the D*reduction program to remove the invented predicates, then the resulting single clause program is one literal shorter than the target program.

## 6 Conclusions and further work

We have introduced the derivation reduction problem (Definition 3.9), the problem of removing derivationally redundant clauses from a clausal theory. We have also introduced a derivation reduction algorithm, which we have used to reduce sets of metarules. We have shown that certain sets of metarules do not have finite reductions (Theorem 4.2), which has implications on completeness not only for MIL, but for any ILP system which relies on metarules. We also compared learning programs using the E-reduced and D-reduced sets of metarules. In general, using the D-reduced set outperforms the E-reduced set both in terms of predictive accuracies and learning times. 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.

*Limitations and future work* Theorem 3.2 shows that the derivation reduction problem is undecidable for general Horn theories. In future work, we wish to study the decidability of the derivation problem for other

fragments of logic, such as function-free theories. For the decidable cases, we wish to identify more efficient reduction algorithms. Theorem 4.2 shows that certain fragments of Datalog do not have finite D-reductions. Future work should explore techniques to mitigate this result, such as exploring whether special metarules, such as a currying metarule [4], could alleviate the issue. We have compared D-reduction to E-reduction, but we would also like to compare other forms of reduction, such as theta-subsumption reduction [27]. We would also like to study other fragments of logic, including triadic logics. We have shown that, although incomplete, the D*-reduced set of metarules outperforms the other sets in our Michalski trains experiment. We would like to explore this incompleteness in more detail, such as determining the degree of incompleteness. Finally, we would like to run more experiments comparing the learning performance of the different sets of metarules on a wider variety of problems.

## Acknowledgements

The authors thank Stephen Muggleton and Katsumi Inoue for helpful discussions on this topic.

## References

1. Aws Albarghouthi, Paraschos Koutris, Mayur Naik, and Calvin Smith. Constraint-based synthesis of datalog programs. In *International Conference on Principles and Practice of Constraint Programming*, pages 689–706. Springer, 2017.
2. Aaron R Bradley and Zohar Manna. *The calculus of computation: decision procedures with applications to verification*. Springer Science & Business Media, 2007.
3. 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.
4. 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.
5. Andrew Cropper and Stephen H. Muggleton. Metagol system. https://github.com/metagol/metagol, 2016.
6. Andrew Cropper and Stephen H. Muggleton. Learning efficient logic programs. *Machine Learning*, pages 1–21, 2018.
7. 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.
8. Richard Evans and Edward Grefenstette. Learning explanatory rules from noisy data. *J. Artif. Intell. Res.*, 61:1–64, 2018.
9. 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.
10. 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.
11. M. R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979.
12. Marijn Heule, Matti Järvisalo, Florian Lonsing, Martina Seidl, and Armin Biere. Clause elimination for sat and qsat. *Journal of Artificial Intelligence Research*, 53:127–168, 2015.
13. Tobias Kaminski, Thomas Eiter, and Katsumi Inoue. Exploiting answer set programming with external sources for meta-interpretive learning. In *34th International Conference on Logic Programming*, 2018.

14. 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.
15. Robert A. Kowalski. Predicate logic as programming language. In *IFIP Congress*, pages 569–574, 1974.
16. Marina Langlois, Dhruv Mubayi, Robert Sloan, and György Turán. Combinatorial problems for horn clauses. *Graph Theory, Computational Intelligence and Thought*, pages 54–65, 2009.
17. J. Larson and Ryszard S. Michalski. Inductive inference of VL decision rules. *SIGART Newsletter*, 63:38–44, 1977.
18. Paolo Liberatore. Redundancy in logic I: CNF propositional formulae. *Artif. Intell.*, 163(2):203–232, 2005.
19. Paolo Liberatore. Redundancy in logic II: 2cnf and horn propositional formulae. *Artif. Intell.*, 172(2-3):265–299, 2008.
20. 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.
21. John W Lloyd. *Foundations of logic programming*. Springer Science & Business Media, 2012.
22. 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.
23. Stephen Muggleton. Inverse entailment and progol. *New Generation Comput.*, 13(3&4):245–286, 1995.
24. 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.
25. 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.
26. Shan-Hwei Nienhuys-Cheng and Ronald de Wolf. *Foundations of Inductive Logic Programming*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1997.
27. G.D. Plotkin. *Automatic Methods of Inductive Inference*. PhD thesis, Edinburgh University, August 1971.
28. 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.
29. Luc De Raedt and Maurice Bruynooghe. Interactive concept-learning and constructive induction by analogy. *Machine Learning*, 8:107–150, 1992.
30. Manfred Schmidt-Schauß. Implication of clauses is undecidable. *Theor. Comput. Sci.*, 59:287–296, 1988.
31. E.Y. Shapiro. *Algorithmic program debugging*. MIT Press, 1983.
32. William Yang Wang, Kathryn Mazaitis, and William W Cohen. Structure learning via parameter learning. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*, pages 1199–1208. ACM, 2014.

# Appendices

## A Exactly-two-connected fragment

**Theorem 4.1** *$\mathcal{E}^2_\infty$ has the same D-reduction as $\mathcal{E}^2_2$.*

*Proof.* Let us consider a clause $C \in \mathcal{E}^2_\infty$ such that $C$ has at least three body literals. If $C$ contains a monadic predicate, e.g. $P(x_1)$, then $x_1$ occurs in only one other literal $P_{x_1}(x_1,x_2)$ (up to the order of the variables and their names) that is dyadic because of the exactly-two-connected nature of $C$ and its length. Replacing in $C$ these two literals by a monadic pivot $P_p(x_2)$ generates a clause $C_1 \in \mathcal{E}^2_\infty$ smaller than $C$ that can be resolved with the clause $C_2$ containing $P(x_1)$, $P_{x_1}(x_1,x_2)$ and $P_p(x_2)$ to derive $C$. Note that if one of $P(x_1)$ or $P_{x_1}(x_1,x_2)$ is the head of $C$ then $P_p(x_2)$ is the head of $C_1$, otherwise, $P_p(x_2)$ is the head of $C_2$.

Let us now assume that $C$ contains no monadic predicate. Then we can pick any variable $x$ such that its two occurrences are in the body of $C$. There must be at least one such variable due to the length of $C$. Let us denote the two literals where $x$ occurs by $P_1(x,x_1)$ and $P_2(x,x_2)$, up to the ordering of the variables in the literals that has no incidence on the proof. Let the clause $C_1$ be $C$ without $\{P_1(x,x_1),P_2(x,x_2)\}$ and with the added pivot literal $P_p(x_1,x_2)$ in the body. This clause belongs to $\mathcal{E}^2_\infty$, is smaller than $C$ and can be resolved with $C_2 = P_p(x_1,x_2) \leftarrow P_1(x,x_1),P_2(x,x_2)$ that also belongs to $\mathcal{E}^2_\infty$ to derive $C$. Hence any clause in $\mathcal{E}^2_\infty \setminus \mathcal{E}^2_2$ can be derived from clauses in $\mathcal{E}^2_2$. Thus any D-reduction of $\mathcal{E}^2_2$ is also a D-reduction of $\mathcal{E}^2_\infty$

## B Two-connected fragment

**Proposition 4.1** *There is no D-reduction of $\mathcal{K}^2_5$ in $\mathcal{K}^2_2$*

*Proof.* Let $C = P_0(x_1,x_2) \leftarrow P_1(x_1,x_3),P_2(x_1,x_4),P_3(x_2,x_3),P_4(x_2,x_4),P_5(x_3,x_4)$, where $C \in \mathcal{K}^2_5$. To derive C from two smaller clauses, these two smaller clauses $C_1$ and $C_2$ must form a partition of the literals in $C$ if one excludes the pivot. Let us consider the possible partitions of $C$. Each of the two sets in the partition must contain at least two elements, otherwise one of $C_1$, $C_2$ would be as big as $C$, which we want to avoid. There are a total of 6 literals in $C$, thus the only partitions of interest are the ones with a 2-4 ratio and a 3-3 ratio, for a total of 25 cases (15 2-4 cases and half of the 20 3-3 cases since we do not care which clause gets which set of literals). Due to the symmetries of the problem, a great number of cases can be safely skipped, reducing the number of cases to consider to only 4.

1. Assume $P_0(x_1,x_2)$ and $P_1(x_1,x_3)$ are the only literals of $C$ that belong to $C_1$. Then, for $C_1$ and $C_2$ to both be in any $\mathcal{K}^2_m$ where $m<5$, the pivot needs to contain at least the variables $x_1$ (occurring only once in $C_2$) and $x_2$, $x_3$ (occurring only once in $C_1$). Since these clauses contain at most dyadic literals, this is not possible. All the 2-4 cases where there is one term variable that connects the two literals occurring in the smaller subset of size two, in either $C_1$ or $C_2$, are symmetrical to this one and thus also impossible to fulfill.
2. Assume $P_0(x_1,x_2)$ and $P_5(x_3,x_4)$ are the only literals of $C$ that belong to $C_1$. Then, the pivot needs to contain all the four variables for $C_1$ and $C_2$ to belong to $\mathcal{K}^2_{m<5}$, which is not possible. All the remaining 2-4 cases are symmetrical to this one.

3. Assume $P_0(x_1,x_2)$, $P_1(x_1,x_3)$ and $P_2(x_1,x_4)$ are the only literals of $C$ in $C_1$. Then, $x_2$, $x_3$ and $x_4$ must occur in the pivot, which is impossible. All the 3-3 cases where the three literals share a common variable are symmetrical.
4. Assume $P_1(x_1,x_2)$, $P_2(x_1,x_4)$ and $P_3(x_2,x_3)$ are the only literals of $C$ in $C_1$. Then, all four variables must occur in the pivot, which is impossible. All the remaining 3-3 cases are symmetrical to this one.

Thus it is not possible to derive $C$ from clauses in $\mathcal{K}^2_2$. For this reason, a D-reduction of $\mathcal{K}^2_5$ cannot be in $\mathcal{K}^2_2$. Note that it cannot be in $\mathcal{K}^2_3$ nor $\mathcal{K}^2_4$ for the same reason.

**Proposition B.1.** *If a clause $C$ is irreducible (i.e. it cannot be derived from clauses of strictly smaller size), has at least two literals in its body and all the term variables it contains occur three times then applying the following transformation on $C$ produces a clause that is also irreducible.*

> *Replace two dyadic literals that share a variable in the body of $C$, denoted as $P_1(x_1,x_2)$ and $P_2(x_1,x_3)$ up to the order of the variables and their names, by the following set of literals: $P_1(x_1,x_4)$, $P_2(x_1,x_5)$, $P_3(x_4,x_5)$, $P_4(x_4,x_2)$, $P_5(x_5,x_3)$ where $P_3$, $P_4$, $P_5$, $x_4$ and $x_5$ are fresh predicate and term variables.*

*Proof.* Let $C$ be a irreducible clause containing the two literals $P_1(x_1,x_2)$ and $P_2(x_1,x_3)$ (without loss of generality) in which all variables occur exactly three times. Let $C_{\text{ext}}$ be the result of the transformation of $C$ where the two previously mentioned literals have been replaced by the set of literals $P_1(x_1,x_4)$, $P_2(x_1,x_5)$, $P_3(x_4,x_5)$, $P_4(x_4,x_2)$, $P_5(x_5,x_3)$ where $P_3$, $P_4$, $P_5$, $x_4$ and $x_5$ are fresh predicate and term variables. Assume that there exist two clauses $C_{\text{ext1}}$ and $C_{\text{ext2}}$ in $\mathcal{K}^2_\infty$ both smaller than $C_{\text{ext}}$, such that $C_{\text{ext1}},C_{\text{ext2}} \vdash C_{\text{ext}}$. If $C_{\text{ext1}}$ is made of a subset of the literals in $C_{\text{ext}} \backslash C$ (plus a pivot), then $C_{\text{ext1}}$ is not two-connected because all these subsets leave three or more variables that occur only once. The corresponding literals for each case are described in Tab. 1 (the symmetrical cases are excluded). To illustrate how the table was built, we consider the case where $C_{\text{ext1}}$ contains $P_1(x_1,x_4)$,$P_2(x_1,x_5)$, $P_3(x_4,x_5)$,$P_4(x_4,x_2)$, i.e. the second line of Tab. 1. Consider these four literals. The variable $x_2$ occurs exactly once. In addition, since the variables $x_1$ and $x_5$ occur only three times in $C_{\text{ext}}$, they also occur only once in $C_{\text{ext2}}$. In the table, variables that occur only once in $C_{\text{ext2}}$ are followed by a star ($\star$). In total, there are three such variables, which is one too many for the pivot to include all of them as arguments. The cases where $C_{\text{ext2}}$ is made only of the literals in $C_{\text{ext}} \backslash C$ plus the pivot are symmetrical to the ones in Tab. 1.

The remaining possibilities are when both $C_{\text{ext1}}$ and $C_{\text{ext2}}$ are made of a mix of the literals in $C_{\text{ext}} \backslash C$ and $C_{\text{ext}} \cap C$. In these cases, the contradiction appears by going from $C_{\text{ext1}},C_{\text{ext2}} \vdash C_{\text{ext}}$ to $C_1,C_2 \vdash C$. For example, if $P_1(x_1,x_4)$ and $P_2(x_1,x_5)$ belong to $C_{\text{ext1}}$ while the other literals from $C_{\text{ext}} \backslash C$ belong to $C_{\text{ext2}}$, then $x_4$ and $x_5$ occur only once in $C_{\text{ext1}}$ (without pivot). There cannot be more than two such literals in the pivot-less $C_{\text{ext1}}$, $C_{\text{ext2}}$ pair or the pivot cannot take all of them as arguments so that they occur at least twice in $C_{\text{ext1}}$ and $C_{\text{ext2}}$ (with pivot), thus $x_4$ and $x_5$ are the only ones. Now consider $C_1$ and $C_2$, obtained respectively from $C_{\text{ext1}}$ and $C_{\text{ext2}}$ by deleting the five literals of $C_{\text{ext}} \backslash C$ from them and adding $P_1(x_1,x_2)$ and $P_2(x_1,x_3)$, i.e. the literals in $C \backslash C_{\text{ext}}$ into $C_1$. Before this transformation, the three occurrences of the variables $x_2$ and $x_3$ were located in $C_{\text{ext2}}$. Due to the deletion of literals, only two occurrences of each remain in $C_2$ and one occurrence of each is now in $C_1$. Hence both $x_2$ and $x_3$ occur only once in that case. Except for the variables $x_4$ and $x_5$ that are absent from $C_1$, $C_2$, the distribution of the remaining variables is unchanged when transforming $C_{\text{ext1}}$, $C_{\text{ext2}}$ in $C_1$, $C_2$, hence these variables occur at least twice. Thus the pair $C_1$ $C_2$ derives $C$ and $C_1$ and $C_2$ are both smaller than $C$, a contradiction.

Table 1: Literals occurring only once in $C_{\text{ext1}}$ or $C_{\text{ext2}}$ when the given literal set is the body of $C_{\text{ext1}}$ - the $\star$ symbol indicates a variable occurring only once in $C_{\text{ext2}}$

| new literals in $C_{\text{ext1}}$ | single-occurrence variables |
|---|---|
| $P_1(x_1,x_4),P_2(x_1,x_5),P_3(x_4,x_5),P_4(x_4,x_2),P_5(x_5,x_3)$ | $x_1\star,x_2\star,x_3\star$ |
| $P_1(x_1,x_4),P_2(x_1,x_5),P_3(x_4,x_5),P_4(x_4,x_2)$ | $x_1\star,x_2,x_5\star$ |
| $P_1(x_1,x_4),P_2(x_1,x_5),P_4(x_4,x_2),P_5(x_5,x_3)$ | $x_1\star,x_2,x_3,x_4\star,x_5\star$ |
| $P_1(x_1,x_4),P_3(x_4,x_5),P_4(x_4,x_2),P_5(x_5,x_3)$ | $x_1,x_2,x_3,x_5\star$ |
| $P_1(x_1,x_4),P_2(x_1,x_5),P_3(x_4,x_5)$ | $x_1\star,x_4\star,x_5\star$ |
| $P_1(x_1,x_4),P_2(x_1,x_5),P_4(x_4,x_2)$ | $x_1\star,x_2,x_4\star,x_5$ |
| $P_1(x_1,x_4),P_3(x_4,x_5),P_4(x_4,x_2)$ | $x_1,x_2,x_5$ |
| $P_1(x_1,x_4),P_3(x_4,x_5),P_5(x_5,x_3)$ | $x_1,x_3,x_4\star,x_5\star$ |
| $P_1(x_1,x_4),P_4(x_4,x_2),P_5(x_5,x_3)$ | $x_1,x_2,x_3,x_4\star,x_5$ |
| $P_3(x_4,x_5),P_4(x_4,x_2),P_5(x_5,x_3)$ | $x_2,x_3,x_4\star,x_5\star$ |
| $P_1(x_1,x_4),P_2(x_1,x_5)$ | $x_1\star,x_4,x_5$ |
| $P_1(x_1,x_4),P_3(x_4,x_5)$ | $x_1,x_4\star,x_5$ |
| $P_1(x_1,x_4),P_4(x_4,x_2)$ | $x_1,x_2,x_4\star$ |
| $P_1(x_1,x_4),P_5(x_5,x_3)$ | $x_1,x_3,x_4,x_5$ |
| $P_3(x_4,x_5),P_4(x_4,x_2)$ | $x_2,x_4\star,x_5$ |
| $P_4(x_4,x_2),P_5(x_5,x_3)$ | $x_2,x_3,x_4,x_5$ |

Table 2: Transformation from $(C_{\text{ext1}},C_{\text{ext2}})$ to $(C_1,C_2)$ and corresponding evolution of the variables occurring only once

| $C_{\text{ext1}}$ ; $C_{\text{ext2}}$ | variables occurring once | $C_1$ ; $C_2$ |
|---|---|---|
| 1,2,3,4,5 ; $\emptyset$ | $\emptyset$ ; $\emptyset$ | 1,2 ; $\emptyset$ |
| 1,2,3,4 ; 5 | $x_5$ ; $x_1$ | 1 ; 2 |
| 1,2,4,5 ; 3 | $x_4,x_5$ ; $\emptyset$ | 1,2 ; $\emptyset$ |
| 1,3,4,5 ; 2 | $x_1,x_5$ ; $\emptyset$ | 1,2 ; $\emptyset$ |
| 1,2 ; 3,4,5 | $x_4,x_5$ ; $x_2,x_3$ | 1,2 ; $\emptyset$ |
| 1,3 ; 2,4,5 | $x_1,x_4,x_5$ ; *** | *** ; *** |
| 1,4 ; 2,3,5 | $x_4$ ; $\emptyset$ | 1 ; 2 |
| 1,5 ; 2,3,4 | $x_1,x_4,x_5$ ; *** | *** ; *** |
| 3,4 ; 1,2,5 | $x_4,x_5$ ; $x_2$ | $\emptyset$ ; 1,2 |
| 4,5 ; 1,2,3 | $x_4,x_5$ ; $x_2,x_3$ | $\emptyset$ ; 1,2 |

By taking into account all the symmetries of the problem, there are only ten such cases to consider. They are summarized in Tab. 2. On the left-hand side of the table is the partition between $C_{\text{ext1}}$ and $C_{\text{ext2}}$ of the five literals in $C_{\text{ext}}\backslash C$. On the right-hand side of the table is the partition between $C_1$ and $C_2$ of the two literals in $C\backslash C_{\text{ext}}$. As was done in the previous example, $C_1$ and $C_2$ are obtained by removing the five literals in $C_{\text{ext}}\backslash C$ from $C_{\text{ext1}}$ and $C_{\text{ext2}}$ respectively and replacing them with the two literals in $C\backslash C_{\text{ext}}$ as indicated in the table. For readability, the literals are only referred to by their number. In the middle of the table are the variables that are known to occur only once in each case (in $C_{\text{ext1}}$ and $C_{\text{ext2}}$ on the left-hand side and in $C_1$ and $C_2$ on the right-hand side). In the cases where there are strictly less than two identified variables that occur only once, there may also be unknown variables that also occur only once, but these are preserved by the transformation

and thus do not impact the reasoning. In most of the cases, it is possible to have at most two literals occurring only once on the right-hand side of the table, implying that $C$ can be derived from two smaller clauses in $\mathcal{K}^2_\infty$, a contradiction. There are also two cases where the assumption that $C_{\text{ext1}}, C_{\text{ext2}} \in \mathcal{K}^2_\infty$ is not verified because there are already more than two variables that occur only once in the explicit parts of $C_{\text{ext1}}$ and $C_{\text{ext2}}$. In such cases, there is nothing to verify so the right-hand side of the table is filled with asterisks (***).

**Theorem 4.2** *$\mathcal{K}^2_\infty$ has no D-reduction.*

*Proof.* The clause $C = P_0(x_1,x_2) \leftarrow P_1(x_1,x_3), P_2(x_1,x_4), P_3(x_2,x_3), P_4(x_2,x_4), P_5(x_3,x_4)$ in $\mathcal{K}^2_5$ was shown irreducible in the proof of Prop. 4.1. It has five literals in its body and all of its term variables occur exactly three times. It can thus be transformed following Prop. B.1 into a bigger irreducible clause in $\mathcal{K}^2_8$. In fact, this transformation preserves all of its requirements (irreducibility, three occurrences of all variables, size of the body greater than three) and can thus be applied iteratively from $C$ so as to generate irreducible clauses in $\mathcal{K}^2_\infty$ that are as big as one wants. For this reason, any D-reduced subset of $\mathcal{K}^2_\infty$ is infinite, thus $\mathcal{K}^2_\infty$ has no D-reduction.