Stratified Negation in RDF Rules: A Correct Approach (Extended Version)*

Nils Küchenmeister[1], Alex Ivliev[1], Dörthe Arndt[2], and Markus Krötzsch[1]

[1] Knowledge-Based Systems Group, TU Dresden, Dresden, Germany
[2] Computational Logic Group, TU Dresden / ScaDS.AI, Dresden/Leipzig, Germany
firstname.lastname@tu-dresden.de

Abstract Combining RDF rule languages, such as N3 or SHACL Rules, with default negation is challenging. Existing methods to stratify negation often fail for RDF rules, since individual triples do not carry enough information to meaningfully restrict potential dependencies. Blank nodes in rule heads further complicate the matter, since the order of rule applications may determine whether new values are created, which in turn can change the applicability of rules with negation. To solve these open problems, we propose *chain stratification* as a robust new condition that guarantees a well-behaved semantics for RDF rules with negation, and existential rules in general. Our condition combines an elaborate analysis of potential multistep derivations with a mechanism for using integrity constraints to discard impossible cases. Applying rules in any order that respects chain stratification is guaranteed to derive an RDF graph that is unique, lean, and justified under the usual negation-as-failure semantics. To show the practicality, we also provide a prototype implementation.

Keywords: Rules languages · N3 · SHACL · Datalog · Full stratification

1 Introduction

The usefulness of rule languages for semantic web query answering, knowledge graph analysis, and ontological reasoning has long been recognised, and various rule engines now support RDF and SPARQL [20,3,13]. Attempts to establish corresponding web standards have been less successful [12,14], but ongoing works on N3 [22] and SHACL Rules [6] may yet change this. Even without standardised syntax, today's rule languages share a common semantic basis — the classical language Datalog [17] — which is intuitive and easy to implement, typically simply by applying rules until nothing new follows.

Some useful features introduce complications though. A first case is that of *non-monotonic* operations, such as *negation as failure* and *aggregation*, the results of which may become invalid in the light of new data. A standard solution to this problem is to *stratify* (i.e., "order") computation, so that non-monotonic

* This is the technical report accompanying our ISWC26 paper [18].

operations are only applied to data that has been fully computed [17]. This precludes feedback cycles from non-monotonic outputs to their own inputs. A second challenge is *value invention* where rules may lead to the creation of new elements, such that an infinite amount of new statements can be derived. In RDF, this is characteristic of rules with *blank nodes* (aka *bnodes*) in their conclusion; in databases and logic, existential quantifiers are used instead [2]. A first mitigation is to create new bnodes only if the rule is not already satisfied by another element in place of the bnode, a method known as *restricted* (or *standard*) *chase* [7]. Non-termination may still occur, and analysing rules to preclude (potential) feedback cycles can help detect such problems [2,4].

The working drafts for N3 [22] and SHACL Rules [6] combine non-monotonic features and value invention with a triple-based data model. Unfortunately, this leads to two major problems: (1) existing stratification and termination analyses fail in many cases, and (2) even if stratification succeeds, the rules may not have a well-defined semantics. Solving these will be our main contribution, but it is useful to first understand each problem in more detail.

Problem 1: Stratification fails with RDF rules. Traditional stratification is based on analysing dependencies between different *predicates* used in the rules. RDF does not have predicates, or, equivalently, has only a single predicate *triple* of arity three,[3] which means that any use of negation prevents stratification. SHACL proposes a refined unification-based dependency analysis [6, Section 3.4 "Stratification"], which helps when triples contain constants, as these N3 rules:[4]

```
1 {?c :participant ?p} => {?p rdf:type :Student}.
2 {?c :teacher ?p . [] log:notIncludes {?p rdf:type :Student}} => {?c :examiner ?p}.
```

The rule in L1 states that every participant `?p` of some course `?c` is a student. The rule in L2 means that course teachers who are not students are also examiners for the course, where `[] log:notIncludes {···}` is N3 syntax for negation. SHACL's stratification method orders L1 before L2, so that the negative precondition is only evaluated after inferring who is a student. Unfortunately, the approach breaks when adding further rules. For example, the following rules *rdfs5* and *rdfs7* are a standard way for handling subproperties in RDFS [10]:

```
3 {?p rdfs:subPropertyOf ?q . ?q rdfs:subPropertyOf ?r} => {?p rdfs:subPropertyOf ?r}.
4 {?p rdfs:subPropertyOf ?q . ?x ?p ?y} => {?x ?q ?y}.
```

The all-variable triples in rule *rdfs7* (L4) create dependencies from and to any other rule, making stratification impossible. Indeed, using all the rules L1–L4, there really are negative feedback cycles on valid input data, e.g., with the triple `:examiner rdfs:subPropertyOf :participant.` In this case, a student might be discovered only after applying rule L2. Similar issues would arise if `:examiner`

[3] Such ternary predicates are also how relational rule engines import RDF [13]. Treating property names as binary predicates to represent RDF rules in Datalog would prevent the use of variables in predicate positions, as required for the RDFS rules.

[4] We use prefixes `rdf:`, `rdfs:`, and `log:` for the standard namespaces [5,22].

were a subproperty of `rdfs:subPropertyOf` or `rdf:type`. These cases are unintended and would likely indicate errors, which N3 can express using *constraints*:

```
5 {:examiner rdfs:subPropertyOf :participant} => false.
6 {:examiner rdfs:subPropertyOf rdfs:subPropertyOf} => false.
7 {:examiner rdfs:subPropertyOf rdf:type} => false.
```

Negative feedback cycles are not possible without violating some of these constraints (which N3 tools would flag as an error), so the use of negation is safe now. Unfortunately, current stratification methods cannot detect this.

Problem 2: Stratified RDF rules lack semantics. Bnodes require the existence of suitable elements, which may or may not need to be created. The next rule, e.g., states that humans must have some biological father who is male:

```
8 {?x rdf:type :Human} => {?x :father _:1 . _:1 rdf:type :Man}.
```

Given (a) `:jo rdf:type :Human`, applying L8 creates a new value for `_:1` (a fresh bnode in RDF, a *labelled null* in databases), but this should not happen if we already have (b) `:jo :father :bob` and (c) `:bob rdf:type :Man`. However, it is not always so easy to tell if a fresh bnode is necessary, and the decision can critically affect the result, as can be shown with some further rules [16, Ex. 6]:

```
9 {?x :father ?y} => {?y rdf:type :Man}.
10 {?x :father ?y} => {?y :eq ?y}.
11 {?x :father ?y1. ?x :father ?y2. [] log:notIncludes {?y1 :eq ?y2}} => {?y1 :nef ?y2}.
```

L9 defines a range for `:father`. L10 derives a simple equality, which is negated in L11 to check for non-equal fathers (`:nef`). Given triples (a) and (b), we could apply L8 to create a fresh bnode `_:a` (`:bob` does not satisfy the conclusion at this point). Rules L9 and L10 would then produce (c), `:bob :eq :bob` and `_:a :eq _:a`, and L11 would yield `:bob :nef _:a` and `_:a :nef :bob`. In contrast, if we apply L9 first, then L8 will not apply, and no `:nef`-triple follows.

The issue is serious: classically stratified RDF rule sets do not have a well-defined semantics. Indeed, L8–L11 are stratified (by SHACL's method), but stratification allows both of the above rule precedences. Other non-monotonic features, such as inequality or aggregation, can also expose the problem.

We might hope to avoid this issue by interleaving rule applications with additional algorithms to find and delete redundant bnodes, creating a *lean* RDF graph [11] (or *core* [7]). But even this (costly) workaround does not solve the problem: we can just apply L9 last. As another solution, Krötzsch proposed a new type of dependency for rules with bnodes (existential variables) in their conclusion [16]. His method can find a rule application order that leads to a lean graph and a well-defined result, but it also fails with rules like rdfs7 (L4).

Our proposed solution. We define a precedence relation on rules that can guide the order of rule applications so that negation-as-failure is justified and the inferred RDF graph is unique and lean (if the input data was). We start from

Krötzsch's *reliances* [16], recalled in the preliminaries (Section 2), which define several types of binary relations between rules to estimate potential interactions.

We generalise this approach based on two observations: (1) the transitive composition of reliances significantly overestimates which (problematic) interactions can occur in real sequences of rule applications, and (2) constraints and plain Datalog rules can effectively rule out problematic cases that would otherwise be allowed in RDF. The difficulty is to turn these into an implementable criterion, in particular (1), since there are infinitely many potential sequences of rules. In Section 3, we introduce *trails* as a data-independent abstraction of such sequences, and show that it gives rise to a rule precedence with desirable properties. Since recognising trails is undecidable, we define *chains* as a decidable relaxation (Section 4), which allows us to introduce *chain stratification* and to show that it meets our requirements (Section 5).

Deciding this new form of stratification remains challenging, since chains can be arbitrarily long. By relating chains to words in a regular language, we finally obtain a decision procedure for chain stratification (Section 6). To show its practical feasibility, Section 8 presents a prototype implementation that builds upon algorithmic methods for reliances [9], and an evaluation that shows applicability to rule sets with thousands of rules. Additional details and proofs can be found in the appendix, and our source code is included in the supplementary material statement at the end of this paper.

2 Preliminaries

We briefly define rules and recall required notions about reliances [16]. We use first-order notation with predicates (not just *triple*) — our results apply to N3 (under common syntax translations [1]) but also to other rule languages [14,13,3].

Rules We use countably infinite, mutually disjoint sets $\boldsymbol{P}$ (*predicates*), $\boldsymbol{V}$ (*variables*), $\boldsymbol{C}$ (*constants*), and $\boldsymbol{N}$ (*nulls*). Each predicate $p \in \boldsymbol{P}$ has an arity $\mathsf{ar}(p) \geq 0$. An *atom* is an expression $p(\boldsymbol{t})$ with $p \in \boldsymbol{P}$ and $\boldsymbol{t} \in (\boldsymbol{V} \cup \boldsymbol{C} \cup \boldsymbol{N})^{\mathsf{ar}(p)}$ a list. A *fact* is a variable-free atom, and a *database* is a set $\mathcal{I}$ of facts. Negation is denoted $\neg$. We often treat lists and conjunctions of (negated) atoms as sets. For any expression or set of expressions E, $\mathsf{var}(E)$ is the set of variables in E.

A *rule* ρ is a null-free expression $\rho : \mathsf{body}^+(\rho) \wedge \mathsf{body}^-(\rho) \rightarrow \exists \boldsymbol{z}.\, \mathsf{head}(\rho)$, where $\boldsymbol{z}$ is a variable list, $\mathsf{body}^+(\rho)$ and $\mathsf{head}(\rho)$ are conjunctions of atoms, and $\mathsf{body}^-(\rho)$ is a conjunction of negated atoms. Variables $\mathsf{var}_\exists(\rho) = \boldsymbol{z}$ are *existential*, all others $\mathsf{var}_\forall(\rho) = \mathsf{var}(\rho) \setminus \boldsymbol{z}$ are *universal* (as usual, we omit their quantifiers). Universal variables in $\mathsf{head}(\rho)$ are *frontier variables*. All universal variables in $\mathsf{head}(\rho)$ and $\mathsf{body}^-(\rho)$ must occur in $\mathsf{body}^+(\rho)$ (*safety*). A rule ρ is *Datalog* if $\mathsf{body}^-(\rho) = \varnothing$ and $\mathsf{var}_\exists(\rho) = \varnothing$. A *ruleset* $\mathcal{R}$ is a finite set of rules; then $\mathcal{R}_D \subseteq \mathcal{R}$ is the subset of Datalog rules. Constraints as in L5–L7 are encoded in Datalog using a nullary head predicate $\bot$. We will discard cases where $\bot$ would be derived in our analyses, but not otherwise endow it with special logical semantics.

Pieces A *piece* ψ of $\mathsf{head}(\rho)$ is a minimal non-empty set $\psi \subseteq \mathsf{head}(\rho)$ such that $\mathsf{var}_\exists(\psi)$ is disjoint from $\mathsf{var}_\exists(\mathsf{head}(\rho) \setminus \psi)$. Let $\mathsf{pieces}(\rho) = \{\exists \mathsf{var}_\exists(\psi).\psi \mid$

ψ a piece of $\mathsf{head}(\rho)\}$, and $\mathsf{pieces}(\mathcal{R}) = \bigcup_{\rho\in\mathcal{R}} \mathsf{pieces}(\rho)$. $\mathcal{R}$ is *piece-decomposed* if each $\rho \in \mathcal{R}$ has just one piece.

Example 1. The rule $a(x) \to \exists v, w.\, b(x), r(x, v), b(v), s(x, w)$ has three pieces: $b(x)$, $\exists v.\, r(x, v), b(v)$ and $\exists w.\, s(x, w)$.

Matches and models A *substitution* is a partial mapping $\sigma : \boldsymbol{V} \to (\boldsymbol{V} \cup \boldsymbol{C} \cup \boldsymbol{N})$. For $t \in \boldsymbol{V} \cup \boldsymbol{C} \cup \boldsymbol{N}$, we set $t\sigma = \sigma(t)$ if $\sigma(t)$ is defined, and $t\sigma = t$ otherwise. Substitutions extend to (sets of) logical expressions as usual. Rule ρ *matches* database $\mathcal{I}$ if there is a substitution $\mu_\forall$ defined on $\mathsf{var}_\forall(\rho)$ such that $\mathsf{body}^+(\rho)\mu_\forall \subseteq \mathcal{I}$ and $\mathsf{body}^-(\rho)\mu_\forall \cap \mathcal{I} = \varnothing$. We denote matches as pairs $\langle\rho, \mu_\forall\rangle$. Match $\langle\rho, \mu_\forall\rangle$ is *satisfied* by $\mathcal{I}$ if there is a substitution $\mu_\exists$ defined on $\mathsf{var}_\exists(\rho)$ such that $\mathsf{head}(\rho)\mu_\forall\mu_\exists \subseteq \mathcal{I}$. $\mathcal{I}$ is a *model* of ρ (or *satisfies* ρ, written $\mathcal{I} \models \rho$) if it satisfies all matches of ρ over $\mathcal{I}$. $\mathcal{I}$ *models* a ruleset $\mathcal{R}$ if $\mathcal{I} \models \rho$ for all $\rho \in \mathcal{R}$, and $\mathcal{I}$ *models* a null-free database $\mathcal{J}$ if $\mathcal{J} \subseteq \mathcal{I}$.

The Chase Algorithms that construct models for a given database and ruleset are called *chase procedures*. We use the following *standard chase*:

Definition 1. *A* chase C *for a null-free database $\mathcal{I}$ and ruleset $\mathcal{R}$ is a (possibly infinite) sequence of databases $\mathsf{db}_C(0), \mathsf{db}_C(1), \ldots$ such that $\mathsf{db}_C(0) = \mathcal{I}$ and:*

(1) for each $i > 0$, there is $\mathsf{rule}_C(i) \in \mathcal{R}$ and $\mathsf{match}_C(i) = \langle\mathsf{rule}_C(i), \mu_\forall\rangle$, such that (a) $\mathsf{match}_C(i)$ is unsatisfied in $\mathsf{db}_C(i-1)$, and (b) $\mathsf{db}_C(i) = \mathsf{db}_C(i-1) \cup \{\mathsf{head}(\mathsf{rule}_C(i))\mu_\forall\mu_\exists\}$ for an injective function $\mu_\exists : \mathsf{var}_\exists(\mathsf{rule}_C(i)) \to \boldsymbol{N}$ that maps variables to distinct fresh nulls (not occurring in $\mathsf{db}_C(i-1)$);
(2) if $\langle\rho, \mu\rangle$ is a match over $\mathsf{db}_C(i)$ for some $i \geq 0$, then there is $j \geq i$ such that $\langle\rho, \mu\rangle$ is satisfied in $\mathsf{db}_C(j)$ (fairness).

The set of chase steps *is* $\mathsf{steps}_C \subseteq \mathbb{N}$. *The result of a chase is* $\mathsf{chase}_C(\mathcal{I}, \mathcal{R}) = \bigcup_{i\in\mathsf{steps}_C} \mathsf{db}_C(i)$. *$C$ is* generating *if every* $\mathsf{match}_C(i)$ *is a match in* $\mathsf{chase}_C(\mathcal{I}, \mathcal{R})$.

In situations like (1), we say $\mathsf{db}_C(i)$ *was obtained by applying* $\mathsf{rule}_C(i)$ *for* $\mu_\forall\mu_\exists$. Many fair chase sequences may exist, based on the choice of rule applied in each step. If C is generating, negated bodies of applied rules remain satisfied. If $\mathcal{R}$ has no existentials, the results of generating chases are exactly the *stable models* [15]. If $\mathcal{R}$ is Datalog, $\mathsf{chase}_C(\mathcal{I}, \mathcal{R})$ is the unique *perfect model*, and we omit $_C$.

Likewise, the chase result is the unique perfect model if $\mathcal{R}$ is *stratified*: Let $\mathsf{preds}(A)$ be the set of predicates used in the set of atoms A, and for $\circ \in \{+, -\}$, let $\prec^\circ_{\mathsf{preds}} := \{\langle\rho_1, \rho_2\rangle \in \mathcal{R}^2 \mid \mathsf{preds}(\mathsf{head}(\rho_1)) \cap \mathsf{preds}(\mathsf{body}^\circ(\rho_2)) \neq \varnothing\}$. Then an existential-free ruleset $\mathcal{R}$ is called (classically) stratified if it can be partitioned into $\mathcal{R} = S_0 \,\dot\cup\, \ldots \,\dot\cup\, S_n$ such that for all $\rho_i \in S_i$ and $\rho_j \in S_j$ we have that $\rho_i \prec^+_{\mathsf{preds}} \rho_j$ implies $i \leq j$, and $\rho_i \prec^-_{\mathsf{preds}} \rho_j$ implies $i < j$. Typically, it is also required that the sets of predicates $\bigcup_{\rho\in S_i} \mathsf{preds}(\mathsf{head}(\rho))$ defined by the strata S_i are disjoint, which is inessential. $\mathcal{R}$ is stratified iff $(\prec^+_{\mathsf{preds}})^\star \circ \prec^-_{\mathsf{preds}}$ is acyclic. A chase exhaustively applying rules from the strata in order is clearly generating.

Things are not so simple with $\exists$ (cf. Section 1, Problem 2).

Universal models and cores Models with nulls are compared using *homomorphisms* (see [16] for a definition). A model $\mathcal{U}$ of $\mathcal{R}$ and $\mathcal{I}$ is *universal*

if all other models $\mathcal{M}$ of $\mathcal{R}$ and $\mathcal{I}$ have a homomorphism to $\mathcal{U}$. Universality is the basis for answering monotonic queries [7]. A *core* is a database without "redundant nulls", which can be defined on finite databases $\mathcal{I}$ by requiring that every homomorphism $\mathcal{I} \to \mathcal{I}$ is surjective (the infinite case is more subtle [16]). The "minimality" of cores makes them useful for non-monotonic queries. A local criterion for a chase to yield a core is based on *alternative matches*:

Definition 2. *Let $\mathcal{I}_a \subseteq \mathcal{I}_b$ be databases, such that $\mathcal{I}_a$ was obtained by applying ρ for μ. A mapping $\mu^A : \mathsf{var}(\rho) \to \boldsymbol{C} \cup \boldsymbol{N}$ is an* alternative match *for $\langle \rho, \mu \rangle$ over $\mathcal{I}_b$ if (1) $\mathsf{head}(\rho)\mu^A \subseteq \mathcal{I}_b$, (2) $x\mu = x\mu^A$ for all $x \in \mathsf{var}_\forall(\rho)$, and (3) there is a null in $\mathsf{head}(\rho)\mu$ that is not in $\mathsf{head}(\rho)\mu^A$. An alternative match in a chase C is one that uses $\mathcal{I}_a = \mathsf{db}_C(i)$, $\rho = \mathsf{rule}_C(i)$, and $\mathcal{I}_b = \mathsf{chase}_C(\mathcal{I}, \mathcal{R})$.*

If a generating chase is finite and has no alternative matches, then it yields a core, but it is undecidable if such a chase exists [16, Thms 4 & 11]. Moreover, rules L8–L11 have generating chases C and C' where all databases $\mathsf{db}_{C^{(\prime)}}(i)$ are cores, but with non-homomorphic (hence non-universal) results. Decidable criteria for unique universal core models can be defined with rule precedences.

Precedences A *precedence* for ruleset $\mathcal{R}$ is a strict partial order $\prec \subseteq \mathcal{R} \times \mathcal{R}$. A chase C *violates* $\prec$ if $\mathsf{rule}_C(i) \prec \mathsf{rule}_C(j)$ for some $i > j$. C *respects* $\prec$ if, for every step i, $\mathsf{db}_C(i)$ has no unsatisfied match $\langle \rho, \mu \rangle$ with $\rho \prec \mathsf{rule}_C(i)$. This is different from "non-violating" since unsatisfied $\langle \rho, \mu \rangle$ may become satisfied without ρ being applied. Krötzsch defines precedences based on three types of *reliance* relations, which were shown to be efficiently computable [9].

Definition 3. *Rule ρ_2* positively relies on *rule ρ_1, written $\rho_1 \prec^+ \rho_2$, if there are databases $\mathcal{I}_a \subseteq \mathcal{I}_b$ such that $\mathcal{I}_b$ was obtained from $\mathcal{I}_a$ by applying ρ_1 for μ_1, and there is an unsatisfied match $\langle \rho_2, \mu_2 \rangle$ over $\mathcal{I}_b$ that is not a match over $\mathcal{I}_a$.*

Definition 4. *Rule ρ_2* negatively relies on *rule ρ_1, written $\rho_1 \prec^- \rho_2$, if there are databases $\mathcal{I}_a \subseteq \mathcal{I}_b$ such that $\mathcal{I}_b$ was obtained from $\mathcal{I}_a$ by applying ρ_1 for μ_1, and there is an unsatisfied match $\langle \rho_2, \mu_2 \rangle$ over $\mathcal{I}_a$ that is not a match over $\mathcal{I}_b$.*

Definition 5. *Rule ρ_1* restrains *rule ρ_2, written $\rho_1 \prec^\square \rho_2$, if there are databases $\mathcal{I}_a \subseteq \mathcal{I}_b$ such that (1) $\mathcal{I}_a$ was obtained by applying ρ_2 for μ_2, (2) $\mathcal{I}_b$ was obtained by applying ρ_1 for μ_1, and (3) there is an alternative match μ^A for $\langle \rho_2, \mu_2 \rangle$ over $\mathcal{I}_b$ that is not an alternative match over $\mathcal{I}_b \setminus \mathsf{head}(\rho_1)\mu_1$.*

The intuition for these notions is that ρ_1 may produce an inference that is needed to match ρ_2 ($\prec^+$), prevents matching ρ_2 ($\prec^-$), or creates an alternative match for ρ_2 ($\prec^\square$). A chase that does not violate $\prec^-$ is generating. A chase that does not violate $\prec^\square$ has no alternative matches.

A ruleset $\mathcal{R}$ is *core stratified* if the relation $\prec^+ \cup \prec^\square$ has no cycles through $\prec^\square$, and *fully stratified* if $\prec^+ \cup \prec^- \cup \prec^\square$ has no cycles through $\prec^- \cup \prec^\square$. For cases where a finite model exists, full stratification (which respects $(\prec^+)^\star \circ (\prec^- \cup \prec^\square)$) yields a generating chase sequence that (if terminating) results in a uniquely determined core model, called the *perfect core model* [16]. The conditions can be relaxed if negation is only used for some predicates [8].

3 Trails: Refining Transitive Positive Reliances

To determine a precedence that yields a perfect core model, it is necessary to estimate when the application of one rule may cause a new unsatisfied match for another rule in the future. This may happen directly, as approximated with $\prec^+$, or possibly involve multiple intermediate steps. Previously, this was achieved by considering the reflexive transitive closure $(\prec^+)^\star$ of positive reliances [16], which is a coarse overestimation. A $\prec^+$-path does not detect whether the sequence of rule applications leading to a match implies that this match is always satisfied. This section aims to improve this by identifying a proper sub-relation of $(\prec^+)^\star$ that records the nuances of such multistep interactions.

To represent sequences of matches, we will use words over an infinite set of *rule instances* (individually denoted by $\hat{\rho}$):

$$\mathsf{instances}(\mathcal{R}) \coloneqq \{\rho\theta \mid \rho \in \mathcal{R}, \theta : \mathsf{var}(\rho) \to (\boldsymbol{V} \setminus \mathsf{var}(\mathcal{R}))\} \tag{1}$$

Example 2. Below, rules ρ_1, ρ_2 and ρ_3 on the left are shown with one of their (infinitely many) instances on the right:

$$\begin{array}{l|l} \rho_1 \colon t(x,y,z), q(x,y) \to p(x,z) & \hat{\rho}_1 \colon t(x_1,y_1,x_1), q(x_1,y_1) \to p(x_1,x_1) \\ \rho_2 \colon p(x,x) \to \exists v.\, s(x,v), t(v,v,v) & \hat{\rho}_2 \colon p(x_1,x_1) \to \exists v_1.\, s(x_1,v_1), t(v_1,v_1,v_1) \\ \rho_3 \colon q(x,y), s(x,z), a(z) \to t(x,x,z) & \hat{\rho}_3 \colon q(x_1,y_1), s(x_1,v_1), a(v_1) \to t(x_1,x_1,v_1) \end{array}$$

The rule instance $\hat{\rho}_1$ was obtained from rule ρ_1 by applying the variable substitution $x \overset{\theta}{\mapsto} x_1$, $y \overset{\theta}{\mapsto} y_1$, $z \overset{\theta}{\mapsto} x_1$, collapsing variables x and z.

We will reuse these rules as a running example throughout Sections 3–5.

Together with a fixed arbitrary bijection $\omega : (\boldsymbol{V} \setminus \mathsf{var}(\mathcal{R})) \to \boldsymbol{C} \cup \boldsymbol{N}$, each instance represents a match. Applying ω to a set of atoms produces a database and, conversely, applying ω^{-1} to a database yields a set of atoms. A sequence of matches $\langle \rho_1, \mu_1 \rangle, \ldots, \langle \rho_n, \mu_n \rangle$ is now represented by a word $\rho_1\mu_1\omega^{-1} \ldots \rho_n\mu_n\omega^{-1}$ over alphabet $\mathsf{instances}(\mathcal{R})$. As the chase algorithm always maps existentials to fresh nulls, we require these words to use fresh variables:

Definition 6. *A finite sequence $(\hat{\rho}_i)_{i=1,\ldots,n} \in \mathsf{instances}(\mathcal{R})^*$ of rule instances $\hat{\rho}_i \in \mathsf{instances}(\mathcal{R})$ with pairwise disjoint existential variables is called* $\exists$-disjoint.

Example 3 (cont.). The sequence of instances $\hat{\rho}_1\hat{\rho}_2\hat{\rho}_3$ is $\exists$-disjoint, as the only existential variable v_1 is fresh in $\hat{\rho}_2$. (It can be used universally again, as in $\hat{\rho}_3$.)

Such a word indicates that the rule application corresponding to its first instance may indirectly cause a new unsatisfied match for the rule whose instance ends the word, if one can give a witnessing chase which applies the matches in the given order, such that each match directly requires the previous one.

Definition 7. *An $\exists$-disjoint sequence of rule instances $t = \hat{\rho}_1, \ldots, \hat{\rho}_n$ is a* trail *if there exists a generating chase sequence C, for which there is a strictly increasing function $f : \{1, \ldots, n\} \to \mathsf{steps}_C$, such that for all $i \in \{1, \ldots, n\}$*

– $\langle \hat{\rho}_i, \omega \rangle = \mathsf{match}_C(f(i))$ *and* *(match correspondence)*
– $i > 1$ *implies* $\mathsf{body}^+(\hat{\rho}_i)\omega \cap \mathsf{head}(\hat{\rho}_{i-1})\omega \neq \varnothing$. *(causal connection)*

We write $\rho_1 \prec_t^+ \rho_2$ *if there is a trail from an instance of* ρ_1 *to an instance of* ρ_2.

Clearly, we have $\prec_t^+ \subseteq (\prec^+)^\star$, as the databases $\mathcal{I}_a = \mathsf{db}_C(f(i) - 1)$ and $\mathcal{I}_b = \mathsf{db}_C(f(i))$ witness the pairwise positive reliances for all i.

Example 4 (cont.). Let $c_{x_1}, c_{y_1} \in \boldsymbol{C}$, $n_{v_1} \in \boldsymbol{N}$, $x_1 \overset{\omega}{\mapsto} c_{x_1}$, $y_1 \overset{\omega}{\mapsto} c_{y_1}$, $v_1 \overset{\omega}{\mapsto} n_{v_1}$. The $\exists$-disjoint sequence $\hat{\rho}_1\hat{\rho}_2\hat{\rho}_3$ is a trail, as there is a chase C with

$\mathsf{db}_C(0) = \{t(c_{x_1}, c_{y_1}, c_{x_1}), q(c_{x_1}, c_{y_1})\}$ $\mathsf{db}_C(1) = \mathsf{db}_C(0) \cup \{p(c_{x_1}, c_{x_1})\}$
$\mathsf{db}_C(2) = \mathsf{db}_C(1) \cup \{s(c_{x_1}, n_{v_1}), t(n_{v_1}, n_{v_1}, n_{v_1})\}$ $\mathsf{db}_C(3) = \mathsf{db}_C(2) \cup \{a(n_{v_1})\}$
$\mathsf{db}_C(4) = \mathsf{db}_C(3) \cup \{t(c_{x_1}, c_{x_1}, n_{v_1})\}$

and step mapping $f : \{1, 2, 3\} \to \{1, 2, 3, 4\}$ with $1 \mapsto 1$, $2 \mapsto 2$, $3 \mapsto 4$, assuming a rule, e.g. $s(x, y) \to a(y)$, to derive fact $a(n_{v_1})$ in step 3. Hence, $\rho_1 \prec_t^+ \rho_3$.

Theorem 1. *If* $\prec_t^+ \circ (\prec^- \cup \prec^\square)$ *is a precedence (hence acyclic), then a chase that respects it does not violate it. In particular, it does not violate* $\prec^-$ *or* $\prec^\square$*, and therefore is generating and free of alternative matches.*

Theorem 1 can be shown by a contrapositive argument, where we take a chase that violates $\prec_t^+ \circ (\prec^- \cup \prec^\square)$, inspect the first violation at chase step i and argue that the chase must disrespect the precedence at i. The other properties follow from slight adaptations of known results on $\prec^-$ and $\prec^\square$ [16].

Theorem 2. *If* $\prec_t^+ \circ (\prec^- \cup \prec^\square)$ *is a precedence, then the result of every chase of* $\mathcal{I}$ *and* $\mathcal{R}$ *that respects it is unique (up to isomorphism).*

Uniqueness is shown by arguing that of two non-isomorphic chases, one must disrespect the precedence. A chase as in Theorem 2 is generating and free of alternative matches by Theorem 1. If finite, this unique result is a core [16, Theorem 11], known as the *perfect core model* of $\mathcal{I}$ and $\mathcal{R}$.

Lemma 3. *The* $\prec_t^+$ *relation is the smallest relation* $<$*, such that respecting* $< \circ \prec$ *ensures non-violation of* $< \circ \prec$ *for acyclic* $< \circ \prec$ *with* $\prec^- \subseteq \prec$.

Based on proof of Theorem 1 outlined above, this is shown by contradiction. Lemma 3 therefore confirms that our choice of $\prec_t^+$ is optimal for our purposes.

Theorem 4. *For* $\hat{\rho}_1, \hat{\rho}_2 \in \mathsf{instances}(\mathcal{R})$*, it is undecidable whether* $\hat{\rho}_1\hat{\rho}_2$ *is a trail.*

Theorem 4 is unsurprising, as chase termination is undecidable. Simulating the run of a Turing-machine with existential rules is a known technique that can be adapted such that the ruleset has a trail iff the TM halts.

Remark 1. As a direct consequence of Theorem 4, it is also undecidable whether there is any trail starting and ending with instances of given rules, since that is already the case for only two instances.

Consequently, we are looking for a decidable relation between $\prec_t^+$ and $(\prec^+)^\star$.

4 Chains: A Decidable Approximation

By Section 3, $\prec_t^+$ is the smallest sub-relation of $(\prec^+)^\star$ that sufficiently captures when one rule application may enable another. We now develop a decidable overestimation of $\prec_t^+$ based on positive reliance checks. When checking $\prec^+$, it suffices to consider instances under ω, which leads to $\prec\!\!\!\prec^+$ as a special case:

Definition 8. *For $\hat{\rho}_1, \hat{\rho}_2 \in \mathsf{instances}(\mathcal{R})$, we write $\hat{\rho}_1 \prec\!\!\!\prec^+ \hat{\rho}_2$, if $\hat{\rho}_1 \prec^+ \hat{\rho}_2$ with $\mathcal{I}_a = (\mathsf{body}^+(\hat{\rho}_1) \cup (\mathsf{body}^+(\hat{\rho}_2) \setminus \mathsf{head}(\hat{\rho}_1)))\omega$, and $\mu_1 = \mu_2 = \omega$.*

Above, database $\mathcal{I}_a$ and matches μ_1 and μ_2 refer to Definition 3.

Example 5 (cont.). It is clear that $\hat{\rho}_1 \prec\!\!\!\prec^+ \hat{\rho}_2$, as $\langle \hat{\rho}_2, \omega \rangle$ is no match on database $\mathcal{I}_a = \{t(c_{x_1}, c_{y_1}, c_{x_1}), q(c_{x_1}, c_{y_1})\}$, but applying the unsatisfied match $\langle \hat{\rho}_1, \omega \rangle$ yields $\mathcal{I}_b = \mathcal{I}_a \cup \{p(c_{x_1}, c_{x_1})\}$ where $\langle \hat{\rho}_2, \omega \rangle$ is an unsatisfied match.

It is natural to ask if a sequence c of instances — as a whole — can cause a new unsatisfied match $\langle \rho, \mu \rangle$, i.e., if it can be *extended* by an instance $\hat{\rho} := \rho\mu\omega^{-1}$. To determine this, we define a *chain rule* ρ_c that is obtained from c by combining all body atoms from the instances as well as all head atoms from all but the last instance $\hat{\rho}_k$ into $\mathsf{body}^+(\rho_c)$ and keeping the head of $\hat{\rho}_k$ as $\mathsf{head}(\rho_c)$. Any match for $\hat{\rho}_k$ that arose due to prior applications of instances $\hat{\rho}_1 \dots \hat{\rho}_{k-1}$ also satisfies the combined list of pre-conditions in $\mathsf{body}^+(\rho_c)$.

Definition 9. *An $\exists$-disjoint sequence of rule instances $c = \hat{\rho}_1, \dots, \hat{\rho}_k$ can be* extended *by $\hat{\rho} \in \mathsf{instances}(\mathcal{R})$ if $\rho_c \prec\!\!\!\prec^+ \hat{\rho}$, where ρ_c is the* chain rule

$$\left(\bigcup_{i \in \{1,\dots,k\}} \mathsf{body}^+(\hat{\rho}_i)\right) \cup \left(\bigcup_{i \in \{1,\dots,k-1\}} \mathsf{head}(\hat{\rho}_i)\right) \to \exists \mathsf{var}_\exists(\hat{\rho}_k).\, \mathsf{head}(\hat{\rho}_k) \quad (2)$$

Definition 10. *An $\exists$-disjoint sequence of rule instances $c = \hat{\rho}_1, \dots, \hat{\rho}_k$ is a* chain *if $k = 1$ or, recursively, $\hat{\rho}_1 \dots \hat{\rho}_{k-1}$ is a chain that can be extended by $\hat{\rho}_k$.*

Example 6 (cont.). To see that $c = \hat{\rho}_1\hat{\rho}_2\hat{\rho}_3$ is a chain, check that $\hat{\rho}_1$ is a chain of length 1, $\rho_{\hat{\rho}_1} \prec\!\!\!\prec^+ \hat{\rho}_2$ (see Example 5), and $\rho_{\hat{\rho}_1\hat{\rho}_2} \prec\!\!\!\prec^+ \hat{\rho}_3$ where the chain rule for $\hat{\rho}_1\hat{\rho}_2$ is $\rho_{\hat{\rho}_1\hat{\rho}_2} : t(x_1, y_1, x_1), q(x_1, y_1), p(x_1, x_1) \to \exists v_1.\, s(x_1, v_1), t(v_1, v_1, v_1)$.

Remark 2. Every trail is a chain.

Although it is easy to determine whether a given sequence of rule instances forms a chain, testing whether there is *any* chain between the instances of two rules is difficult, as chains can in principle grow arbitrarily long. We can, however, further restrict to certain chains:

Definition 11. *For a sequence $s = \hat{\rho}_1, \dots, \hat{\rho}_n$, the set of* stale variables *is:*

$$\mathsf{stale}(s) := \bigcup_{i,j \in \{1,\dots,n\}.\, j<i} ((\mathsf{var}_\forall(\hat{\rho}_i) \setminus \mathsf{var}(\mathsf{head}(\hat{\rho}_{i-1}))) \cap \mathsf{var}_\forall(\hat{\rho}_j)) \quad (3)$$

If $\mathsf{stale}(s) = \varnothing$, then s is decoupled.

Decoupled chains are special because the only variables re-used in the ith instance are frontier and existential variables stemming from the $(i-1)$th instance.

Example 7 (cont.). The sequence $t = \hat{\rho}_1\hat{\rho}_2\hat{\rho}_3$ is not decoupled, as y_1 is re-used in $\hat{\rho}_3$ although not present in $\mathsf{var}(\mathsf{head}(\hat{\rho}_2)) = \{x_1, v_1\}$ and thus $\mathsf{stale}(t) = \{y_1\} \neq \varnothing$.

This shows that trails are not always decoupled. We can, however, relate each trail to a decoupled chain.

Definition 12. *A sequence $\hat{\rho}_1^A, \ldots, \hat{\rho}_n^A$* generalises *a sequence $\hat{\rho}_1^B, \ldots, \hat{\rho}_n^B$ if there is a variable substitution $\sigma : \boldsymbol{V} \to \boldsymbol{V}$, such that $\hat{\rho}_i^A\sigma = \hat{\rho}_i^B$ for all $i \in \{1, \ldots, n\}$.*

Example 8 (cont.). Any sequence of rule instances can be rewritten such that it is decoupled, by injectively replacing the stale variables with fresh ones. Consider $\theta_1 = \theta_2 = \mathrm{id}$ and θ_3 with $y_1 \mapsto y_2$. Then $c = (\hat{\rho}_1\theta_1)(\hat{\rho}_2\theta_2)(\hat{\rho}_3\theta_3)$ is decoupled, and c generalises t with $\sigma = \theta_1^{-1} \circ \theta_2^{-1} \circ \theta_3^{-1}$, which simply replaces y_2 by y_1.

Lemma 5. *Every trail can be generalised by a decoupled chain.*

Lemma 5 is shown by naming the trail's stale variables apart, along the lines of Example 8. The resulting word may not be a trail any more, but is necessarily a chain. To show $\rho_{\hat{\rho}_1\theta_1\ldots\hat{\rho}_{k-1}\theta_{k-1}} \not\prec^+ \hat{\rho}_k\theta_k$ for all word positions k, we consider $\mathcal{I}_a$ and $\mathcal{I}_b$ obtained from the original trail's witnessing chase and augmented with additional copies of facts under suitable renamings $\omega^{-1}\theta_1\omega$ up to $\omega^{-1}\theta_k\omega$.

Example 9 (cont.). Database $\mathcal{I}_a$ from Definition 8 will, in addition to $q(c_{x_1}, c_{y_1})$, contain a fact $q(c_{x_1}, c_{y_2})$ matching $\mathsf{body}^+(\hat{\rho}_3\theta_3)$ with named apart variable y_2.

For ruleset $\mathcal{R}$, we collect all decoupled chains in set $\mathsf{chains}(\mathcal{R}) \subseteq \mathsf{instances}(\mathcal{R})^*$, which may be infinite. We write $\rho_1 \prec_c^+ \rho_2$ if there is a decoupled chain between instances of ρ_1 and ρ_2. With Lemma 5, we know that indeed $\prec_t^+ \subseteq \prec_c^+$, hence:

Corollary 6 (of Theorem 1). *A chase that respects $\prec_c^+ \circ (\prec^- \cup \prec^\square)$ does not violate $\prec^-$ or $\prec^\square$, and therefore is generating and free of alternative matches.*

However, as the following example shows, there are decoupled chains that do not generalise any trail, i.e. $\prec_t^+ \subset \prec_c^+$ (the containment is proper).

Example 10 (cont.). Recall that—in order to give the witnessing chase for the trail $t = \hat{\rho}_1\hat{\rho}_2\hat{\rho}_3$ in Example 4—we had to assume the existence of a rule to derive a fact $a(n_{v_1})$ after applying instance $\hat{\rho}_2$, which invented the null n_{v_1}. This fact cannot originate from the input database, as it pertains to a null. If, however, the full ruleset were $\mathcal{R} = \{\rho_1, \rho_2, \rho_3\}$, t would not be a trail (but still a chain).

If $c = \hat{\rho}_1, \ldots, \hat{\rho}_n$ is a chain, then $\hat{\rho}_i \prec^+ \hat{\rho}_{i+1}$ for $i \in \{1, \ldots, n-1\}$, but the reverse does not hold, as attested below: The inclusion $\prec_c^+ \subset (\prec^+)^\star$ is proper.

Example 11. Consider the rules $\rho_1 : r(x,y), a(y) \to b(x)$, $\rho_2 : b(x) \to c(x)$, and $\rho_3 : c(x) \to \exists v.\, r(x,v), a(v)$, which represent common types of ontological axioms. They are pairwise positively relying, i.e. $\rho_1 \prec^+ \rho_2 \prec^+ \rho_3$, but there are no instances thereof that would form a chain, as the way ρ_1 derives $b(x)$ ensures that $\mathsf{head}(\rho_3)$ is already satisfied for any match of ρ_3 introduced by applying ρ_1 and then ρ_2. Therefore, the relation $\prec_c^+$ is indeed a proper sub-relation of $(\prec^+)^\star$.

In Section 5 we further refine the composition of $\prec_c^+$ with $\prec^\square$ and $\prec^-$, and use it to introduce chain stratification. Afterwards, in Section 6, we examine how $\prec_c^+$ can be computed despite the fact that chains could be arbitrarily long.

5 Rule Selection for Chain-Stratified Rulesets

Having established that respecting $\prec_c^+ \circ (\prec^- \cup \prec^\square)$ prevents violation of $\prec^-$ and $\prec^\square$, we now take a closer look at the negative or restraint reliance after the chain's last instance: We introduce relations $\prec_c^- \subset \prec_c^+ \circ \prec^-$ and $\prec_c^\square \subset \prec_c^+ \circ \prec^\square$, and use them to demarcate a class of rulesets which we call (fully) chain-stratified. These subsume fully stratified rulesets [16].

Definition 13. *A chain $c = \hat{\rho}_1 \dots \hat{\rho}_n$ restrains a rule ρ if $\rho_c \prec^\square \rho$ where ρ_c is the chain rule of c. If $\hat{\rho}_1$ is an instance of $\rho_1 \in \mathcal{R}$, then we write $\rho_1 \prec_c^\square \rho$.*

Clearly, $\rho_1 \prec_c^\square \rho$ implies $\rho_1 \prec_c^+ \circ \prec^\square \rho$, but the reverse does not hold:

Example 12 (cont. of Ex. 10). The last instance $\hat{\rho}_3$ restrains both rules $\rho_4 : p(x,x) \to \exists v. t(v,x,v)$ and $\rho_5 : q(x,y) \to \exists v.\, t(x,y,v)$ but ρ_c only restrains ρ_5.

Respecting $\prec_c^\square$ still prevents $(\prec_c^+ \circ \prec^\square)$-violations causing alternative matches.

Lemma 7. *Let $t = \hat{\rho}_1 \dots \hat{\rho}_n$ be a trail and C a witnessing chase for t with step mapping f. If there is a chase step $i < f(n)$, such that $\mathsf{rule}_C(i)$ has an alternative match over $\mathsf{db}_C(f(n))$ but not over $\mathsf{db}_C(f(n)-1)$, then $\hat{\rho}_1 \prec_c^\square \mathsf{rule}_C(i)$.*

This is shown by obtaining a pair of databases from the offending chase and applying Definition 5. Negative reliances are treated similarly:

Definition 14. *A rule ρ negatively relies on a chain $c = \hat{\rho}_1 \dots \hat{\rho}_n$ if $\rho_c \prec^- \rho$. If $\hat{\rho}_1$ is an instance of $\rho_1 \in \mathcal{R}$, then we write $\rho_1 \prec_c^- \rho$.*

Lemma 8. *Let $t = \hat{\rho}_1 \dots \hat{\rho}_n$ be a trail and C a witnessing chase for t with step mapping f. If there is a chase step $i < f(n)$ such that $\mathsf{match}_C(i)$ is a match for $\mathsf{rule}_C(i)$ over $\mathsf{db}_C(f(n)-1)$ but not over $\mathsf{db}_C(f(n))$, then $\hat{\rho}_1 \prec_c^- \mathsf{rule}_C(i)$.*

With the above improvements, we can finally define our novel stratification condition, which we call *chain stratification*:

Definition 15. *A ruleset $\mathcal{R}$ is* core chain-stratified *if $\langle \mathcal{R}, \prec_c^\square \rangle$ is acyclic. It is* (fully) chain-stratified *if $\langle \mathcal{R}, \prec_c^- \cup \prec_c^\square \rangle$ is acyclic.*

Corollary 9 (of Theorems 1 and 2, and Lemma 5). *On a chain-stratified ruleset the result of any chase that respects $\prec_c^- \cup \prec_c^\square$ is generating, alternative match-free, and unique (up to isomorphism). If finite, it is a perfect core model.*

For chain-stratified rulesets, the rule precedence $\prec_c^- \cup \prec_c^\square$ induces a "stratification" where rules may belong to multiple strata, obtained by first computing the minimum-rank layering $L_0 \dots L_n$ of the precedence, and then forming strata $S_i = \bigcup_{j \leq i} L_j$. Successively chasing fixpoints for $S_0 \dots S_n$ ensures that no new unsatisfied matches for any $(\prec_c^- \cup \prec_c^\square)$-predecessors of S_i are added.

6 The Regular Language of Chains

We now address the issue of deciding whether there is an (arbitrarily long) decoupled chain connecting instances of two given rules. Enumerating the set $\mathsf{chains}(\mathcal{R})$ merely yields a semi-decision procedure, because a ruleset can have infinitely many chains. Therefore, we characterise $\mathsf{chains}(\mathcal{R})$ with a language $\mathcal{L}_\mathcal{R}$ and establish that $\mathcal{L}_\mathcal{R}$ is regular, which yields a decision procedure. Our restriction to *decoupled* chains (Section 4) is essential for this to work.

We assign to each $c \in \mathsf{chains}(\mathcal{R})$ a label $\ell(c) \in \Sigma$ from some (yet to be defined) alphabet Σ. Intuitively, $\ell(c)$ captures all information needed for reliance checks between the chain rule ρ_c and any rule from $\mathcal{R}$, but in bounded space that does not grow arbitrarily with (the body of) ρ_c. This requires some auxiliary notation. For ruleset $\mathcal{R}$, the finite set $\mathsf{variants}(\mathcal{R}) \coloneqq \{\rho\theta \mid \rho \in \mathcal{R}, \theta : \mathsf{var}(\rho) \to \mathsf{var}(\mathcal{R})\}$ is obtained by replacing variables in $\mathcal{R}$ by any combination of variables found in $\mathcal{R}$. Further, let $\mathsf{orig} : \mathsf{instances}(\mathcal{R}) \to \mathsf{variants}(\mathcal{R})$ be an arbitrary but fixed mapping such that $\hat{\rho} = \mathsf{orig}(\hat{\rho})\theta_{\mathsf{orig}(\hat{\rho})}$ with injective variable renaming $\theta_{\mathsf{orig}(\hat{\rho})}$.

Example 13 (cont. of Ex. 2). Let $\mathcal{R} = \{\rho_1, \rho_2, \rho_3\}$. Then $\mathsf{var}(\mathcal{R}) = \{x, y, z, v\}$ and, e.g., $\check{\rho}_1^a : t(v, z, y), q(v, z) \to p(v, y)$ and $\check{\rho}_1^b : t(x, y, x), q(x, y) \to p(x, x)$ both are variants of ρ_1. In our example instance $\hat{\rho}_1$ of rule ρ_1, the variables in the first and the third position of predicate t are collapsed. So we can set $\mathsf{orig}(\hat{\rho}_1) = \check{\rho}_1^b$ with the injection $\theta_{\mathsf{orig}(\hat{\rho}_1)} : x \mapsto x_1, y \mapsto y_1$, which yields $\hat{\rho}_1 = \mathsf{orig}(\hat{\rho}_1)\theta_{\mathsf{orig}(\hat{\rho}_1)}$.

Moreover, given a set $\mathcal{F}$ of formulae $\exists \boldsymbol{v}. \psi$ where ψ is a conjunction of atoms, let $\mathsf{parts}(\mathcal{F}) \coloneqq \{\exists \tilde{\boldsymbol{v}}. \tilde{\psi} \mid \exists \boldsymbol{v}. \psi \in \mathcal{F}, \tilde{\psi} \subseteq \psi, \tilde{\psi} \neq \varnothing, \tilde{\boldsymbol{v}} = \boldsymbol{v} \cap \mathsf{var}(\tilde{\psi})\}$.

Example 14. Consider the set $\mathsf{parts}(\mathsf{pieces}(\mathsf{variants}(\mathcal{R})))$ of "partial pieces of rule variants." The rule $a(x) \to \exists v. r(x, v), b(v)$ has a single piece, and contributes three partial pieces: (1) $\exists v. r(x, v), b(v)$, (2) $\exists v. r(x, v)$, and (3) $\exists v. b(v)$. Another variant of the same rule is $a(x) \to r(x, x), b(x)$, which has two partial pieces $r(x, x)$ and $b(x)$ (but not $r(x, x), b(x)$, which is no piece).

To understand how we label a chain c, consider the chain rule ρ_c and some rule $\rho \in \mathcal{R}$. To check $\rho_c \prec^+ \rho$, Definition 3 requires $\mathcal{I}_a$ and $\mathcal{I}_b$ such that: (i) the positive body of ρ matches $\mathcal{I}_b$ but not $\mathcal{I}_a$ (i.e., ρ_c contributed something relevant to $\mathcal{I}_b$); (ii) the negative body of ρ is not matched in $\mathcal{I}_b$; and (iii) applying

ρ adds something over $\mathcal{I}_b$ (match unsatisfied). To check these, we store three components: (C) pieces of ρ_c that contribute new facts when applying ρ_c ($\rightsquigarrow$ (i)); (S) partial pieces satisfied after applying ρ_c ($\rightsquigarrow$ (iii)); and (N) variants of rules inhibited by negative body atoms after applying ρ_c ($\rightsquigarrow$ (ii)).

Now we can define the finite alphabet $\Sigma \coloneqq 2^{\mathcal{H}} \times 2^{\mathcal{F}} \times 2^{\mathsf{variants}(\mathcal{R})}$ where $\mathcal{H} \coloneqq \mathsf{pieces}(\mathsf{variants}(\mathcal{R}))$ and $\mathcal{F} \coloneqq \mathsf{parts}(\mathcal{H})$. To define the *labelling function* ℓ, consider a chain $c = \hat{\rho}_1 \dots \hat{\rho}_n$ with chain rule ρ_c, and let $\theta_n \coloneqq \theta_{\mathsf{orig}(\hat{\rho}_n)}$, i.e., $\hat{\rho}_n = \mathsf{orig}(\hat{\rho}_n)\theta_n$. Moreover, let $\mathcal{I}_c \coloneqq \big(\mathsf{body}^+(\rho_c) \cup \mathsf{head}(\rho_c)\big)\,\theta_n^{-1}\omega$, which represents the necessary facts in $\mathcal{I}_b$ when checking $\rho_c \prec^+ \rho$ for any $\rho \in \mathcal{R}$ (Definition 3), in the notation of Definition 8 with the fixed mapping ω. We define $\ell(c) = \langle \ell^{\mathsf{C}}(c), \ell^{\mathsf{S}}(c), \ell^{\mathsf{N}}(c) \rangle$ where:

$$\ell^{\mathsf{C}}(c) \coloneqq \{\exists \boldsymbol{v}.\tilde{\psi} \in \mathsf{pieces}(\mathsf{orig}(\hat{\rho}_n)) \mid \tilde{\psi}\theta_n \not\subseteq \mathsf{body}^+(\rho_c)\} \tag{4}$$

$$\ell^{\mathsf{S}}(c) \coloneqq \{\exists \tilde{\boldsymbol{v}}.\tilde{\psi} \in \mathcal{F} \mid \mathcal{I}_c \models \exists \tilde{\boldsymbol{v}}.\tilde{\psi}\omega_\forall\} \tag{5}$$

$$\ell^{\mathsf{N}}(c) \coloneqq \{\check{\rho} \in \mathsf{variants}(\mathcal{R}) \mid \big(\mathcal{I}_c \cup \mathsf{body}^+(\check{\rho})\omega\big) \cap \mathsf{body}^-(\check{\rho})\omega \neq \varnothing\} \tag{6}$$

where $\omega_\forall$ is the restriction of ω to universal variables. Note that $\ell^{\mathsf{C}}(c)$ contains all head pieces of ρ_c except possibly Datalog atoms. The main correctness property of ℓ is this: if $\ell(c_1) = \ell(c_2)$ for chains c_1 and c_2, then for all $\rho \in \mathcal{R}$ and all $\prec \in \{\prec^+, \prec^-, \prec^\square\}$, we have that $\rho_{c_1} \prec \rho$ iff $\rho_{c_2} \prec \rho$. For $\prec^+$, the proof of this claim establishes that the three label components can indeed be used as in the intuition given above. Cases for $\prec^-$ and $\prec^\square$ are similar. We can then show:

Lemma 10. *Let $c_1, c_2 \in \mathsf{chains}(\mathcal{R})$ with $\ell(c_1) = \ell(c_2)$. If $c_1\hat{\rho}_1 \in \mathsf{chains}(\mathcal{R})$ then there is $\hat{\rho}_2 \in \mathsf{instances}(\mathcal{R})$ (both $\hat{\rho}_1$ and $\hat{\rho}_2$ belong to the same $\rho \in \mathcal{R}$), such that*

$$\text{(i) } c_2\hat{\rho}_2 \in \mathsf{chains}(\mathcal{R}) \qquad \textit{and} \qquad \text{(ii) } \ell(c_1\hat{\rho}_1) = \ell(c_2\hat{\rho}_2)\ .$$

Part (i) largely follows from the correctness for $\prec^+$ by Definitions 9 and 10. Part (ii) is easy to see for ℓ^{C} and ℓ^{N}; for ℓ^{S}, it holds because $\mathcal{F}$ is closed under $\mathsf{parts}(\mathsf{variants}(\cdot))$ and the instances added to the end of the chains are decoupled.

The *word function* is $w : \mathsf{chains}(\mathcal{R}) \to \Sigma^*$ with $c \mapsto \ell(c)$ for $|c| = 1$, $c \mapsto w(c_{\mathsf{pre}})\ell(c)$ for $c = c_{\mathsf{pre}}\hat{\rho}_n$. The *language of chains* is $\mathcal{L}_\mathcal{R} = \{w(c) \mid c \in \mathsf{chains}(\mathcal{R})\}$. Now given two words $u, v \in \mathcal{L}_\mathcal{R}$ with $u_{|u|} = v_{|v|}$, $uw \in \mathcal{L}_\mathcal{R}$ implies $vw \in \mathcal{L}_\mathcal{R}$ for any $w \in \Sigma^*$, which follows from an inductive application of Lemma 10. An equivalence relation on words based purely on their last letter therefore fully characterises their *extensions* into longer words. By Myhill-Nerode, we find that:

Theorem 11. *The language $\mathcal{L}_\mathcal{R}$ is regular.*

Algorithm 1 outlines a procedure that uses these insights to check chain stratification. We represent chains $c = \hat{\rho}_1 \dots \hat{\rho}_n$ as 4-tuples $\langle \rho_1, \rho_n, \rho_c, w(\hat{\rho}_1 \dots \hat{\rho}_{n-1}) \rangle$. We start with single-rule chains (line A4) for rules in the range of $\prec^-$ or $\prec^\square$; this suffices since we search for $\prec_c^- \cup \prec_c^\square$-cycles and $\prec_c^- \cup \prec_c^\square$ only ranges over such rules. We then iteratively extend $\mathcal{C}$ and $\prec$ (A5) until a cycle is found (A6) or all chains have been considered and no cycle found (A17). For each $\rho_c \prec^+ \rho$, we consider all concrete mappings η such that $\rho_c\eta \not\prec^+ \hat{\rho}\eta$ (A9), if the η-specific

Algorithm 1: Is ruleset $\mathcal{R}$ chain-stratified?

1 $\mathcal{G}_{\mathsf{rel}} \coloneqq \langle \mathcal{R}, \prec^+, \prec^-, \prec^\square \rangle$; `// compute the reliance graph`
2 **if** $\mathcal{G}_{\mathsf{rel}}$ *has no cycle via* $\prec^- \cup \prec^\square$ **then : return** "$\mathcal{R}$ is fully stratified" ;
3 $\prec \coloneqq \prec^- \cup \prec^\square$; `// precedence from` $\mathcal{G}_{\mathsf{rel}}$ `(will approach` $\prec_c^- \cup \prec_c^\square$`)`
4 $\mathcal{C} \coloneqq \{\langle \rho, \rho, \rho, \epsilon \rangle \mid \rho \in \mathcal{R}$ and there is $\rho' \in \mathcal{R}$ with $\rho' \prec^- \rho$ or $\rho' \prec^\square \rho$ in $\mathcal{G}_{\mathsf{rel}}\}$;
5 **for** $\langle \rho_1, \rho_n, \rho_c, w(c_{\mathsf{pre}}) \rangle \in \mathcal{C}$ **and** $\rho \in \mathcal{R}$ **with** $\rho_n \prec^+ \rho$ *in* $\mathcal{G}_{\mathsf{rel}}$ **and** $\rho_c \prec^+ \rho$ **:**
6 **if** $\prec$ *is cyclic* **then : return** "$\mathcal{R}$ is not chain stratified";
7 $\hat{\rho} \coloneqq$ injectively rename ρ, such that it does not use $\mathsf{var}(\rho_c)$;
8 $V \coloneqq \mathsf{var}(\mathsf{head}(\rho_c) \cup \mathsf{body}^+(\hat{\rho}))$; $C \coloneqq \mathsf{symbols}(\mathsf{head}(\rho_c) \cup \mathsf{body}^+(\hat{\rho}))$;
9 **for** $\eta : V \to V \cup C$ **with** $\rho_c \eta \not\prec^+ \hat{\rho}\eta$ **:**
10 **if** $\ell(c\eta) \in w(c_{\mathsf{pre}})$ **then : Continue**;
11 $\rho_{c\eta\hat{\rho}\eta} \coloneqq (\mathsf{body}^+(\rho_c) \cup \mathsf{head}(\rho_c) \cup \mathsf{body}^+(\hat{\rho}))\eta \to \exists \mathsf{var}_\exists(\hat{\rho})\eta.\, \mathsf{head}(\hat{\rho})\eta$;
12 $\mathcal{C} \coloneqq \mathcal{C} \cup \{\langle \rho_1, \rho, \rho_{c\eta\hat{\rho}\eta}, w(c_{\mathsf{pre}})\ell(c\eta) \rangle\}$;
13 **for** $\rho' \in \mathcal{R}$ **with** $\rho \prec^\square \rho'$ *in* $\mathcal{G}_{\mathsf{rel}}$ **:**
14 **if** $\rho_{c\eta\hat{\rho}\eta} \prec^\square \rho'$ **then :** $\prec \coloneqq \prec \cup \{\langle \rho_1, \rho' \rangle\}$;
15 **for** $\rho' \in \mathcal{R}$ **with** $\rho \prec^- \rho'$ *in* $\mathcal{G}_{\mathsf{rel}}$ **:**
16 **if** $\rho_{c\eta\hat{\rho}\eta} \prec^- \rho'$ **then :** $\prec \coloneqq \prec \cup \{\langle \rho_1, \rho' \rangle\}$;
17 **return** "$\mathcal{R}$ is chain stratified";

chain's label was not encountered yet (A10). Since chains are extended stepwise, presence of a label in $w(c_{\mathsf{pre}})$ means a label-equivalent chain was already considered (and all related restraints and negative reliances found). Then we construct the extended chain rule (A11), add the new chain (A12, this adds to the cases iterated in A5), and add new pairs to $\prec$ (A13–16).

Keeping a hash set of seen chain labels and checking if labels of new chains are already present alleviates the need to store $w(c)$ as a whole. Chain labels can even be canonised, as Lemma 10 generalises to chains with isomorphic labels.

7 Further Improving Chain Stratification

Chain stratification generalises full stratification, which controls the interaction of value invention and negation, and solves, e.g., Problem 2 (L8–L11) of Section 1. Some generalisations, outlined next, allow us to cover more cases, e.g., Problem 1.

Closure under constraints All reliances require a minimal set of facts (up to homomorphism), e.g., $\mathcal{I}_a$ in Def. 3. If the Datalog rules $\mathcal{R}_D \subseteq \mathcal{R}$ entail $\bot$ on these facts, then the reliance can be discarded [19]. For chains c, we could apply $\mathcal{R}_D$ to $\mathcal{I}_c$ (see Section 6), but for Algorithm 1 to be correct, we also must ensure that the label (not the specific chain) determines if $\bot$ is derived or not.

We therefore consider additional formulae $\mathcal{B}$ to capture partial matches of Datalog rule bodies: $\mathcal{B} \coloneqq \mathsf{parts}(\{\exists \boldsymbol{v}.\, \mathsf{body}^+(\check{\rho}) \mid \check{\rho} \in \mathsf{variants}(\mathcal{R})_D, \boldsymbol{v} \subseteq \mathsf{var}_\forall(\check{\rho})\})$. In $\ell(c)$, we will replace $\ell^{\mathsf{S}}(c) \subseteq \mathcal{F}$ (5) by $\ell^{\mathsf{S}}_D(c) \subseteq \mathcal{F} \cup \mathcal{B}$. We convert sets $S \subseteq \mathcal{F} \cup \mathcal{B}$ to databases $\mathsf{toDb}(S) \coloneqq \bigcup \{\tilde{\psi}\omega_\forall \mu_\exists \mid \exists \tilde{\boldsymbol{v}}.\tilde{\psi} \in S, \mu_\exists : v \mapsto \mathsf{fresh}(v, \exists \tilde{\boldsymbol{v}}.\tilde{\psi})\}$, using the mapping ω and an auxiliary injection $\mathsf{fresh} : \mathsf{var}(S) \times S \to \boldsymbol{N} \setminus (\mathsf{var}(\mathcal{R})\omega)$.

For chain $c = \hat{\rho}_1 \dots \hat{\rho}_n$ with $\theta_i \coloneqq \theta_{\mathsf{orig}(\hat{\rho}_i)}$, we recursively define $\ell^{\mathsf{S}}_D(c)$, where we use $\mathsf{Pre}_c \coloneqq \varnothing$ if $n = 1$ and $\mathsf{Pre}_c \coloneqq \mathsf{toDb}(\ell^{\mathsf{S}}_D(\hat{\rho}_1 \dots \hat{\rho}_{n-1}))\omega^{-1}\theta_{n-1}$ otherwise:

$$\ell^{\mathsf{S}}_D(c) \coloneqq \{\exists\tilde{\boldsymbol{v}}.\tilde{\psi} \in \mathcal{F} \cup \mathcal{B} \mid \mathsf{chase}((\mathsf{Pre}_c \cup \mathsf{body}^+(\hat{\rho}_n) \cup \mathsf{head}(\hat{\rho}_n))\theta_n^{-1}\omega, \mathcal{R}_D) \models \exists\tilde{\boldsymbol{v}}.\tilde{\psi}\omega_\forall\} \tag{7}$$

Compared to (5), we replaced $\mathcal{I}_c$ by a chase result based on the previous chain label and the new final rule, which has a homomorphism to $\mathsf{chase}(\mathcal{I}_c, \mathcal{R}_D)$. Relying on the previous label lets us lift Lemma 10 and related correctness results to ℓ^{S}_D. We also recast the extended label as a rule:

$$\rho^D_c : \mathsf{toDb}(\ell^{\mathsf{S}}_D(c))\omega^{-1}\theta_n \setminus \mathsf{head}(\hat{\rho}_n) \to \exists\mathsf{var}_\exists(\hat{\rho}_n).\, \mathsf{head}(\hat{\rho}_n) \tag{8}$$

An $\exists$-disjoint sequence c can be *extended* by $\hat{\rho}$ *under constraints* if $\rho^D_c \not\prec^+ \hat{\rho}$. It is a *chain* if $\bot \notin \mathsf{body}^+(\rho^D_c)$ and Definition 10 holds. We write $\rho_1 \prec^{\square D}_c \rho_2$ (respectively $\rho_1 \prec^{-D}_c \rho_2$) if there is c with $\rho^D_c \prec^{\square} \rho_2$ (respectively $\rho^D_c \prec^{-} \rho_2$).

Definition 16. *$\mathcal{R}$ is* chain-stratified under constraints *if $\prec^{-D}_c \cup \prec^{\square D}_c$ is acyclic.*

Definition 16 generalises chain stratification since only fewer chains are considered. Correctness is retained if the chase prioritises Datalog rules:

Corollary 12. *If $\mathcal{R}$ is chain-stratified under constraints, a chase that respects $\prec_D \coloneqq \prec^{-D}_c \cup \prec^{\square D}_c \cup (\mathcal{R}_D \times (\mathcal{R} \setminus \mathcal{R}_D))$ is generating and alternative match-free.*

The relation $\prec_D$ is acyclic if $\prec^{-D}_c \cup \prec^{\square D}_c$ is, since $\prec_D$ ranges over $\mathcal{R} \setminus \mathcal{R}_D$.

Example 15. We return to Problem 1 (L1–L7), denoting the respective rules $\rho_1, \dots, \rho_7$. For $\mathcal{R} = \{\rho_1, \rho_2, \rho_3, \rho_4\}$, we find reliances $\rho_1 \prec^- \rho_2$, $\rho_3 \prec^+ \rho_3$, $\rho_4 \prec^- \rho_2$, and $\rho \prec^+ \rho_4 \prec^+ \rho$ for all $\rho \in \mathcal{R}$ (due to the all-variable triple `?x ?p ?y` in L4). $\mathcal{R}$ is not fully stratified, e.g., due to $\rho_2 \prec^+ \rho_4 \prec^+ \rho_1 \prec^- \rho_2$.

$\mathcal{R}$ is not chain stratified either. For clarity, we substitute constants from the image of ω directly. To construct a chain for $\rho_2\rho_4\rho_1$, consider instances $\hat{\rho}_2 = \rho_2$, $\hat{\rho}_4 = \rho_4$[`?p/:examiner,?q/:participant,?x/?c,?y/?p`], and $\hat{\rho}_1 = \rho_1$. Then $\hat{\rho}_2$ is a length-1 chain, and $\hat{\rho}_2\hat{\rho}_4$ is a chain since $\hat{\rho}_2 \not\prec^+ \hat{\rho}_4$, with chain rule $\rho_{2,4}$:

```
{?c :teacher ?p . :examiner rdfs:subPropertyOf :participant . ?c :examiner ?p}
    => {?c :participant ?p}.
```

We have $\rho_{2,4} \not\prec^+ \hat{\rho}_1$, so $\hat{\rho}_2\hat{\rho}_4\hat{\rho}_1$ is a chain, hence $\rho_2 \prec^+_c \rho_1$. Similarly, $\rho_{2,4,1} \prec^- \rho_2$, so we have a cycle $\rho_2 \prec^-_c \rho_2$, and $\mathcal{R}$ is not chain stratified.

However, the extended $\mathcal{R} = \{\rho_1, \dots, \rho_7\}$ is chain stratified under constraints, where $\mathcal{R}_D = \mathcal{R} \setminus \{\rho_2\}$. The $\prec^+$-paths from ρ_2 to ρ_1 that do not revisit ρ_2 or pass through ρ_1 can be described by a regular expression $\rho_2(\rho_4\rho_3^*)^*\rho_4\rho_1$, and each such path corresponds to one or more chains. Similarly, $\rho_2(\rho_4\rho_3^*)^*\rho_4$, for ρ_2 to ρ_4. All chains from ρ_2 to ρ_1 violate constraints ρ_5 or ρ_6, and the chains from ρ_2 to ρ_4 that negatively rely on ρ_2 violate constraint ρ_7. The above chain $\hat{\rho}_2\hat{\rho}_4\hat{\rho}_1$ will be discarded due to ρ_5. Since `rdfs:subPropertyOf` is transitive by ρ_3, longer

chains of the form $\rho_2\rho_4^+\rho_1$ entail the same constraint-violating fact. Any alternative chain from ρ_2[?p/rdf:type] to ρ_1 via ρ_4[?q/rdfs:subPropertyOf], entails ?c rdfs:subPropertyOf rdf:type. Then any triple ?x ?c :Student entails ?x rdf:type :Student, which could create a negative feedback cycle for ρ_2. Such chains are discarded due to ρ_6. Similarly, any chain starting like $\rho_2\rho_4^+\rho_3\ldots$ violates ρ_6. All surviving chains of the form $\rho_2\rho_4^*\rho_4$[?q/rdf:type], whose chain rules negatively rely on ρ_2, are eliminated by ρ_7. The remaining chains (all of the form $\rho_2\rho_4^+$) are unproblematic.

Considering chains is still crucial here. For a "full stratification under constraints" (following [19]), we would need to prevent $\rho_2 \prec^+ \rho_4$ with a stronger constraint such as {:examiner rdfs:subPropertyOf ?q} => false, which would forbid useful triples, e.g., :examiner rdfs:subPropertyOf :teacher.

Null awareness When building chain rules as in (2) (or (8) under constraint closure), the information which variables are existential is lost for all but the last instance. To prohibit them from unifying with constants, we can restrict a fixed arbitrary injection $\boldsymbol{V} \to \boldsymbol{N}$ to $\mathsf{var}_\exists(\hat{\rho}_{k-1})$ and apply it to the chain rule. The latter may thus contain nulls, which can eliminate some irrelevant reliances.

Negation awareness Keeping negations of the atoms $\bigcup_{i\in\{1,\ldots,k\}} \mathsf{body}^-(\hat{\rho}_i)$ in the chain rule's body sharpens the analysis further. They may prevent some outgoing reliances of the chain rule. However, this needs to be reflected in $\ell^{\mathsf{N}}(c)$, tracking the subset of these atoms that only use frontier variables. This is clearly bounded. It suffices, as only such atoms may unify with facts of the representative databases examined for reliance checks between the chain rule and other rules.

8 Implementation & Evaluation

We implement null- and negation-aware chain stratification under $\mathcal{R}_D$-closure in a Rust library[5] and a PoC tool using it, which accepts Nemo [13] and VLog [21] syntax, as well as (a subset of) N3 syntax (via a Python transpiler).

While our method is designed to address known problematic cases that arise in RDF rules, there are no representative RDF-based rulesets that could be used to evaluate its *effectiveness*. However, we can investigate the feasibility of our analysis in terms of *performance*. To this end, we use our tool to check chain stratification on a benchmark that was also used by González *et al.* [9] in their analysis of core stratification, which consists of 201 piece-decomposed, negation-free rulesets with predicate symbols (not just triple).[6] Without negation, core and full chain stratification coincide, but the effort of computing chains is still realistic. We ran our experiments on a Linux server (2×QuadCore Intel Xeon 3.5GHz, 768GiB RAM), with a 15min timeout per analysis. Each run is repeated thrice and the median time values are reported in our supplementary material.

[5] Source code (ca. 13k LoC) available in the supplementary material statement.
[6] They correspond to a subset of the *Oxford Ontology Repository*, accessed 2025-12-01.

Overall, 80 rulesets were classified in less than 0.1sec, 115 in less than 1sec, 160 in under 1 min, and 187 within the total timeout. The remaining 14 rulesets that did not finish in that time each contained over 60,000 rules. 128 cases were not core-stratified (the previous best condition for negation-free rules), and required the analysis of chains. These cases on average required an additional 12.51% of analysis time in comparison to the check for core stratification. Considering the relative complexity of our definitions, we consider this to be a very acceptable overhead that appears to be feasible in practice, especially since many sets of RDF rules are not covered by any other conditions.

9 Conclusion

We established chain stratification to improve beyond prior notions. Augmented by constraints, our approach enables the use of N3 rules like *rdfs7* in combination with blank nodes and negation. We designed a refined criterion based on the transitive closure of positive reliances, contributed interesting theoretical results on their decidability, and provided a prototypical implementation in Rust. Promising directions of future work may include (1) integration with RDF rule engines, (2) repair of non-chain-stratified rulesets by mining constraints from counter examples generated by reliance computations, (3) transfer of our finding to the handling of aggregates, and (4) analysis of the utility of $\prec_c^+$ to establish new termination criteria. Finally, we hope that our insights can also contribute to ongoing standardisation activities of RDF rules.

Acknowledgments. This work is supported by Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) in project number 389792660 (TRR 248, Center for Perspicuous Systems) and 390696704 (CeTI Cluster of Excellence), by the Bundesministerium für Forschung, Technologie und Raumfahrt (BMFTR, Federal Ministry of Research, Technology and Space) in the Center for Scalable Data Analytics and Artificial Intelligence (ScaDS.AI), and in DAAD project 57616814 (SECAI, School of Embedded Composite AI) as part of the program Konrad Zuse Schools of Excellence in Artificial Intelligence.

Supplemental Material Statement. The source code of our prototypical Rust implementation is published at https://gitlab.com/nkuechen/chaining-reliances/-/tree/iswc26. Refer to `README.md` to reproduce the results.

Declaration of use of Generative AI. During the preparation of this work the authors used ChatGPT to ask for Rust programming tips. Opus 4.6 was used for proofreading. The text in this paper as well as the source code is fully handwritten.

References

1. Arndt, D., Mennicke, S.: Existential Notation3 Logic. Theory Pract. Log. Program. **25**(3), 304–339 (2025). https://doi.org/10.1017/S1471068425000055, https://doi.org/10.1017/s1471068425000055

2. Baget, J.F., Leclère, M., Mugnier, M.L., Salvat, E.: On rules with existential variables: Walking the decidability line. Artificial Intelligence **175**(9–10), 1620–1654 (2011)
3. Bellomarini, L., Sallinger, E., Gottlob, G.: The Vadalog system: Datalog-based reasoning for knowledge graphs. Proc. VLDB Endowment **11**(9), 975–987 (2018). https://doi.org/10.14778/3213880.3213888
4. Cuenca Grau, B., Horrocks, I., Krötzsch, M., Kupke, C., Magka, D., Motik, B., Wang, Z.: Acyclicity notions for existential rules and their application to query answering in ontologies. J. of Artificial Intelligence Research **47**, 741–808 (2013)
5. Cyganiak, R., Wood, D., Lanthaler, M. (eds.): RDF 1.1 Concepts and Abstract Syntax. W3C Recommendation (25 February 2014), available at http://www.w3.org/TR/rdf11-concepts/
6. David, R., Habgood, D., Seaborne, A., Steyskal, S. (eds.): SHACL 1.2 Rules. W3C Working Draft (02 April 2026), available at https://www.w3.org/TR/2026/WD-shacl12-rules-20260402/
7. Deutsch, A., Nash, A., Remmel, J.B.: The chase revisited. In: Lenzerini, M., Lembo, D. (eds.) Proc. 27th Symp. on Principles of Database Systems (PODS'08). pp. 149–158. ACM (2008)
8. Ellmauthaler, S., Krötzsch, M., Mennicke, S.: Answering queries with negation over existential rules. In: Proc. 36th AAAI Conf. on Artificial Intelligence (AAAI'22). pp. 5626–5633 (2022). https://doi.org/10.1609/aaai.v36i5.20503
9. González, L., Ivliev, A., Krötzsch, M., Mennicke, S.: Efficient dependency analysis for rule-based ontologies. In: Sattler, U., Hogan, A., Keet, M., Presutti, V., Almeida, J.P.A., Takeda, H., Monnin, P., Pirrò, G., d'Amato, C. (eds.) Proc. 21st International Semantic Web Conference (ISWC 2022). LNCS, vol. 13489, pp. 267–283. Springer (2022). https://doi.org/10.1007/978-3-031-19433-7_16
10. Hayes, P., Patel-Schneider, P.F. (eds.): RDF 1.1 Semantics. W3C Recommendation (25 February 2014), available at http://www.w3.org/TR/rdf11-mt/
11. Hogan, A.: Canonical forms for isomorphic and equivalent RDF graphs: Algorithms for leaning and labelling blank nodes. ACM Trans. Web **11**(4), 22:1–22:62 (2017). https://doi.org/10.1145/3068333
12. Horrocks, I., Patel-Schneider, P.F., Boley, H., Tabet, S., Grosof, B.N., Dean, M.: SWRL: A Semantic Web Rule Language. W3C Member Submission (21 May 2004), available at http://www.w3.org/Submission/SWRL/
13. Ivliev, A., Gerlach, L., Meusel, S., Steinberg, J., Krötzsch, M.: Nemo: Your friendly and versatile rule reasoning toolkit. In: Marquis, P., Ortiz, M., Pagnucco, M. (eds.) Proc. 21st Int. Conf. on Principles of Knowledge Representation and Reasoning (KR'24). pp. 743–754. IJCAI Organization (2024). https://doi.org/10.24963/kr.2024/70
14. Kifer, M., Boley, H. (eds.): RIF Overview. W3C Working Group Note (22 June 2010), available at http://www.w3.org/TR/rif-overview/
15. Konczak, K., Linke, T., Schaub, T.: Graphs and colorings for answer set programming. Theory Pract. Log. Program. **6**(1-2), 61–106 (2006)
16. Krötzsch, M.: Computing cores for existential rules with the standard chase and ASP. In: Calvanese, D., Erdem, E., Thielscher, M. (eds.) Proc. 17th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR'20). pp. 603–613. IJCAI (2020). https://doi.org/10.24963/kr.2020/60
17. Krötzsch, M.: Modern Datalog: Concepts, methods, applications. In: Artale, A., Bienvenu, M., García, Y.I., Murlak, F. (eds.) Joint Proceedings of the 20th and 21st Reasoning Web Summer Schools (RW 2024 & RW 2025). OASIcs, vol. 138. Dagstuhl Publishing (2025). https://doi.org/10.4230/OASIcs.RW.2024/2025.7

18. Küchenmeister, N., Ivliev, A., Arndt, D., Krötzsch, M.: Stratified negation in RDF rules: A correct approach. In: Koubarakis, M., Vidal, M.E., Polleres, A., van Erp, M., Ruiz, E.J., Seneviratne, O., Aroyo, L., Demartini, G., Alharbi, R., Barile, R., d'Amato, C., Tamma, V. (eds.) Proc. 25th International Semantic Web Conference (ISWC 2026). LNCS, vol. tbd, p. tbd. Springer (2026). https://doi.org/tbd
19. Magka, D., Krötzsch, M., Horrocks, I.: Computing stable models for nonmonotonic existential rules. In: Rossi, F. (ed.) Proc. 23rd Int. Joint Conf. on Artificial Intelligence (IJCAI'13). pp. 1031–1038. AAAI Press/IJCAI (2013)
20. Nenov, Y., Piro, R., Motik, B., Horrocks, I., Wu, Z., Banerjee, J.: RDFox: A highly-scalable RDF store. In: et al., M.A. (ed.) Proc. 14th Int. Semantic Web Conf. (ISWC'15), Part II. LNCS, vol. 9367, pp. 3–20. Springer (2015). https://doi.org/10.1007/978-3-319-25010-6_1
21. Urbani, J., Jacobs, C., Krötzsch, M.: Column-oriented Datalog materialization for large knowledge graphs. In: Schuurmans, D., Wellman, M.P. (eds.) Proc. 30th AAAI Conf. on Artificial Intelligence (AAAI'16). pp. 258–264. AAAI Press (2016). https://doi.org/10.1609/aaai.v30i1.9993
22. Van Woensel, W., Arndt, D., Champin, P.A., Tomaszuk, D., Kellogg, G. (eds.): Notation3 Language. W3C Draft Community Group Report (15 May 2024), available at https://w3c.github.io/N3/spec/

A Proofs for Section 3

In the ensuing proofs, we will use the following additional notation:

- $\mathsf{matches}_{\mathcal{R}}(\mathcal{I})$ for the set of all matches of rules from $\mathcal{R}$ over database $\mathcal{I}$.
- $\mathsf{unsat}_{\mathcal{R}}(\mathcal{R}) \subseteq \mathsf{matches}_{\mathcal{R}}(\mathcal{I})$ to collect all the unsatisfied matches over $\mathcal{I}$.
- $\mathsf{hom}_C(k)$ for $\mu_\forall\mu_\exists$ in the extended match $\mathsf{match}_C(k) = \langle \mathsf{rule}_C(k), \mu_\forall\mu_\exists\rangle$ applied in step k of chase C.

Theorem 1. *If $\prec_t^+ \circ (\prec^- \cup \prec^\square)$ is a precedence (hence acyclic), then a chase that respects it does not violate it. In particular, it does not violate $\prec^-$ or $\prec^\square$, and therefore is generating and free of alternative matches.*

We will show a slightly stronger version of the theorem for $\prec_t^+ \circ \prec$, assuming only that $\prec^- \subseteq \prec$. (The specifics of $\prec^\square$ are irrelevant for the proof.)

Proof (of contrapositive of Theorem 1). Let C be a chase sequence that violates $\prec$. Then there must be chase steps $i \leq k$ with $\mathsf{rule}_C(k) \prec \mathsf{rule}_C(i)$. In fact, $i < k$ as there would otherwise be a self-loop in $\prec_c^+ \circ \prec$. Select such $\langle i, k\rangle$ (lexicographic) minimally.

Define the following sequence of indices:

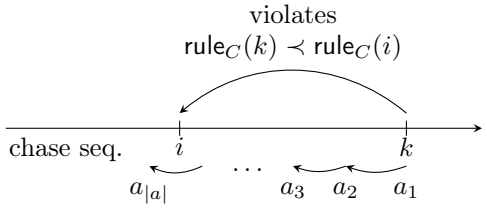
$$a_1 \coloneqq k \tag{9}$$

$$a_{x+1} \coloneqq \min\{j \mid \mathsf{match}_C(a_x) \in \mathsf{matches}_{\mathcal{R}}(\mathsf{db}_C(j))\} \qquad \text{if } i < a_x \tag{10}$$

This sequence is clearly strictly decreasing (and finite). Also, it has length $|a| > 1$. For all $x < |a|$, we have $i < a_x$. Only $a_{|a|}$ may be smaller or equal to i.

Due to the minimality of $\langle i, k\rangle$, and because of $\prec^- \subseteq \prec$, no negative reliance can be violated in C up to step k. Therefore, C is generating up to k. Hence, we know that a match selected in step a_x that was already available in step $j < a_x$ must also be available in all intermediate steps.

The second-to-last match $\mathsf{match}_C(a_{|a|-1})$ was already applicable in step $a_{|a|} \leq i$. We therefore clearly have $\mathsf{match}_C(a_{|a|-1}) \in \mathsf{matches}_{\mathcal{R}}(\mathsf{db}_C(i))$.



The sequence of rule instances $t = \hat{\rho}_1, \ldots, \hat{\rho}_{|a|}$ with

$$\hat{\rho}_x = \mathsf{rule}_C(a_{|a|+1-x})\mathsf{hom}_C(a_{|a|+1-x})\omega^{-1} \qquad \text{for } x \in \{1, \ldots, |a|\} \tag{11}$$

is a trail, because the chase C with step mapping $f : \{1, \ldots, |a|\} \to \mathsf{steps}_C$, $x \overset{f}{\mapsto} a_{|a|+1-x}$ fulfils the two conditions from Definition 7. Match correspondence

follows from the definition of $\hat{\rho}_x$ based on the matches applied in step $f(x)$ of C. Causal connection is a consequence of the subsequent sequence item a_{x+1} being the minimal chase step where the match used at a_x was available. This means that $\mathsf{match}_C(a_x)$ must map a body atom of $\mathsf{rule}_C(a_x)$ onto a fact added by applying $\mathsf{match}_C(a_{x+1})$, i.e. $\mathsf{head}(\hat{\rho}_x) \cap \mathsf{body}^+(\hat{\rho}_{x+1}) \neq \varnothing$.

We thus have $\mathsf{rule}_C(a_{|a|}) \prec_t^+ \mathsf{rule}_C(a_1)$.

As we set $a_1 = k$, we have $\mathsf{rule}_C(a_1) = \mathsf{rule}_C(k)$ and we hence $\mathsf{rule}_C(a_1) \prec \mathsf{rule}_C(i)$. This gives $\mathsf{rule}_C(a_{|a|}) \prec_t^+ \circ \prec \mathsf{rule}_C(i)$ and both rules were selectable in chase step i. We also know that $\mathsf{match}_C(a_{|a|}) \neq \mathsf{match}_C(i)$, because $\prec_t^+ \circ \prec$ would otherwise have a self-loop. Hence, C disrespects $\prec_t^+ \circ \prec$, as $\mathsf{match}_C(a_{|a|})$ should have been preferred over $\mathsf{match}_C(i)$ in chase step i. □

Theorem 2. *If $\prec_t^+ \circ (\prec^- \cup \prec^\square)$ is a precedence, then the result of every chase of $\mathcal{I}$ and $\mathcal{R}$ that respects it is unique (up to isomorphism).*

Proof (of contrapositive of Theorem 2). Let C_1 and C_2 be two generating chases of (the same) $\langle \mathcal{I}, \mathcal{R} \rangle$ with non-isomorphic results. This can only be the case if there is (w.l.o.g) a match that is applied in C_1 but never applied in C_2. Further, there must be such a match, whose application in C_1 has added facts which are not present in C_2 (‡). Select a minimal $i \in \mathsf{steps}_{C_1}$, such that $\mathsf{match}_{C_1}(i)$ is such a match. All matches applied before i in C_1 must therefore be applied at some point in C_2. Let $i_2 \in \mathsf{steps}_{C_2}$ be the chase step at which the last of them was applied. The database $\mathsf{db}_{C_1}(i-1) = \mathcal{I}_0 \cup \bigcup_{k<i} \mathsf{head}(\mathsf{rule}_{C_1}(k))\mathsf{hom}_{C_1}(k)$ must homomorphically map into $\mathsf{db}_{C_2}(i_2)$ (†). Thus, any unsatisfied match over $\mathsf{db}_{C_2}(i_2)$ must either (1) also be unsatisfied over $\mathsf{db}_{C_1}(i-1)$ or (2) not be a match there (because the necessary facts were not derived yet). In case (2), it must be derivable by a sequence of applications starting with an unsatisfied match over $\mathsf{db}_{C_1}(i-1)$. The $\mathsf{match}_{C_1}(i)$ may either be (a) unsatisfied, (b) satisfied, or (c) invalidated[7] over $\mathsf{db}_{C_2}(i_2)$. If (a), then, there must be a $k \geq i_2$, such that $\mathsf{match}_{C_1}(i)$ is no unsatisfied match over $\mathsf{db}_{C_2}(k)$ (due to fairness). Applying $\mathsf{match}_{C_2}(k)$ has either (b) satisfied $\mathsf{match}_{C_1}(i)$, or (c) invalidated it. In case (b), $\mathsf{rule}_{C_1}(i)$ cannot be Datalog, as satisfying it would otherwise not have added facts in C_1 which C_2 lacks (as is required by ‡). Either $\mathsf{rule}_{C_1}(i)$ has both Datalog and existential pieces, which makes it self-restraining. This would mean that C_1 does not respect the precedence $\prec^\square \subseteq \prec_t^+ \circ (\prec^- \cup \prec^\square)$. Or $\mathsf{rule}_{C_1}(i)$ has only existential pieces, which means that there is an alternative match for $\mathsf{match}_{C_1}(i)$ over $\mathsf{db}_{C_2}(k)$, i.e. $\mathsf{rule}_{C_2}(k) \prec^\square \mathsf{rule}_{C_1}(i)$ by Definition 5. In case (c), $\mathsf{rule}_{C_2}(k) \prec^- \mathsf{rule}_{C_1}(i)$ by Definition 4. Similar to the proof of Theorem 1 above, we construct a sequence of chase steps from C_2 like

$$a_1 \coloneqq k \tag{12}$$

$$a_{x+1} \coloneqq \min\{j \mid \mathsf{match}_{C_2}(a_x) \in \mathsf{matches}_{\mathcal{R}}(\mathsf{db}_{C_2}(j))\} \tag{13}$$

It is finite, as it is clearly strictly decreasing and all sequence items are non-negative. In fact, it ends with $a_{|a|} = 0$. By (†) and because C_2 is generating, we

[7] by deriving a fact from $\mathsf{body}^-(\mathsf{rule}_{C_1}(i))\mathsf{hom}_{C_1}(i)$

must be able to find an x, such that either (1) $\mathsf{match}_{C_2}(a_x) \in \mathsf{unsat}_{\mathcal{R}}(\mathsf{db}_{C_1}(i-1))$, or (2) there exists a $\langle \rho, \mu \rangle \in \mathsf{unsat}_{\mathcal{R}}(\mathsf{db}_{C_1}(i-1))$ with appropriate step mapping for C_2 witnessing that $t_{\mathsf{pre}} = \rho\mu\omega^{-1} \dots \mathsf{rule}_{C_2}(a_x)\mathsf{hom}_{C_2}(a_x)\omega^{-1}$ is a trail. We then use $a_x \dots a_1$ to define a sequence of rule instances t and step mapping f. By Definition 7, C_2 and f witness that t is a trail. So we now have that $\mathsf{rule}_{C_2}(a_x) \prec_t^+ \mathsf{rule}_{C_2}(k)(\prec^- \cup \prec^{\square})\mathsf{rule}_{C_1}(i)$. If (1), there is an unsatisfied match over $\mathsf{db}_{C_1}(i-1)$ for $\mathsf{rule}_{C_2}(a_x)$ (and clearly also for $\mathsf{rule}_C(i)$). As $\prec_t^+ \circ (\prec^- \cup \prec^{\square})$ is acyclic by assumption, $\mathsf{rule}_C(i) \neq \mathsf{rule}_{C_2}(a_x)$. So C_1 does not respect $\prec_t^+ \circ (\prec^- \cup \prec^{\square})$, as $\mathsf{rule}_{C_2}(a_x)$ should have been preferred over $\mathsf{rule}_{C_1}(i)$. If (2), $\rho \prec_t^+ \mathsf{rule}_{C_2}(a_x) \prec_t^+ \mathsf{rule}_{C_2}(k)(\prec^- \cup \prec^{\square})\mathsf{rule}_{C_1}(i)$. The sequence $t_{\mathsf{pre}}t$ is also a trail, i.e. $\rho \prec_t^+ \mathsf{rule}_{C_2}(k)(\prec^- \cup \prec^{\square})\mathsf{rule}_{C_1}(i)$ and $\langle \rho, \mu \rangle$ should have been preferred over $\mathsf{rule}_{C_1}(i)$. Again, C_1 disrespects $\prec_t^+ \circ (\prec^- \cup \prec^{\square})$. □

Lemma 3. *The $\prec_t^+$ relation is the smallest relation $<$, such that respecting $< \circ \prec$ ensures non-violation of $< \circ \prec$ for acyclic $< \circ \prec$ with $\prec^- \subseteq \prec$.*

Proof. Let $< \subset \prec_t^+$ be a smaller relation. Then there must be some $\rho_1 \prec_t^+ \rho_2$ with $\rho_1 \not< \rho_2$. Consider a relation $\prec \supseteq \prec^-$, such that there is some ρ_3, such that $\rho_2 \prec \rho_3$ and $< \circ \prec$ is acyclic. Suppose towards contradiction that any chase respecting $< \circ \prec$ has no $\prec$ violations. Consider a chase sequence C that violates $\rho_2 \prec \rho_3$, but no other $\prec$-edges. Analogous to the proof of Theorem 1, select $i < k$ minimally, such that $\mathsf{rule}_C(k) \prec \mathsf{rule}_C(i)$ (in fact, $\mathsf{rule}_C(k) = \rho_2$ and $\mathsf{rule}_C(i) = \rho_3$), then construct the sequence $(a_x)_{x=1,\dots|a|}$, and the trail $t = \rho_1, \dots, \rho_{|a|}$. In chase step i, both $\mathsf{match}_C(i)$ and $\mathsf{match}_C(a_{|a|})$ are selectable, but not $\mathsf{match}_C(k)$. As $\rho_1 \not< \rho_2$, we do not have $\mathsf{rule}_C(a_|a|) = \rho_1 < \circ \prec \rho_3 = \mathsf{rule}_C(i)$, so C respects $< \circ \prec$. □

Theorem 4. *For $\hat{\rho}_1, \hat{\rho}_2 \in \mathsf{instances}(\mathcal{R})$, it is undecidable whether $\hat{\rho}_1\hat{\rho}_2$ is a trail.*

Proof (by reduction from the halting problem). Let M be a Turing machine with initial state q_0 and single "accept" state q_f and let w be a word. It is undecidable whether M has a finite run on w that reaches the "accept" state.

- Turing machine $M = \langle Q, \Sigma, \Gamma, \delta, q_0, q_f \rangle$ with set of states Q, input alphabet Σ, tape alphabet $\Gamma \supseteq \Sigma$, (partial) transition function $\delta : Q \times \Gamma \to Q \times \Gamma \times \{L, R\}$, initial state q_0 and final state q_f.
- Elements of $\Gamma^* \times Q \times \Gamma^*$ are configurations of M. For $\delta(q, a) = \langle q', a', d \rangle$, write $\langle xa, q, by \rangle \vdash \langle xa'b, q', y \rangle$ iff $d = R$ and $\langle xa, q, by \rangle \vdash \langle x, q', a'by \rangle$ iff $d = L$.
- For word $w \in \Sigma^*$, a run of M is $\langle w, q_0, \epsilon \rangle \vdash \dots$. If this run is finite and ends with q_f, then M accepts w.

Encode M and w into a ruleset $\mathcal{R}(M, w)$ with two designated rules $\rho_1, \rho_2 \in \mathcal{R}(M, w)$, such that $\rho_1 \prec_t^+ \rho_2$ if and only if M holds and accepts w. In detail:

Input: $\langle M, w\rangle$ for word $w \in \Sigma^*$
Output: $\mathcal{R}(M, w)$ (constants in bold)

$$\begin{aligned}
\rho_1 :\quad & \to \exists t_0, x_1, \ldots, x_{|w|}.\, \mathit{state}(t_0, \boldsymbol{q_0}), \mathit{pos}(t_0, x_1), \mathit{tape}(t_0, x_1, \boldsymbol{w_1}), \mathit{next}(x_1, x_2),\\
& \qquad \ldots, \mathit{tape}(t_0, x_{|w|-1}, \boldsymbol{w_{|w|-1}}), \mathit{next}(x_{|w|-1}, x_{|w|}), \mathit{tape}(t_0, x_{|w|}, \boldsymbol{w_{|w|}})\\
& \mathit{state}(t, s) \to \exists t'.\, \mathit{step}(t, t')\\
& \mathit{tape}(t, x, c) \to \exists x'.\, \mathit{next}(x, x')\\
& \mathit{tape}(t, x, c), \mathit{pos}(t, x'), x \neq x', \mathit{step}(t, t') \to \mathit{tape}(t', x, c)\\
& \mathit{state}(t, \boldsymbol{s}), \mathit{pos}(t, x), \mathit{tape}(t, x, \boldsymbol{c}), \mathit{next}(x', x), \mathit{step}(t, t')\\
& \qquad \to \mathit{state}(t', \boldsymbol{s'}), \mathit{tape}(t', x, \boldsymbol{c'}), \mathit{pos}(t', x') \qquad \text{for } \delta(\boldsymbol{s}, \boldsymbol{c}) = \langle \boldsymbol{s'}, \boldsymbol{c'}, L\rangle\\
& \mathit{state}(t, \boldsymbol{s}), \mathit{pos}(t, x), \mathit{tape}(t, x, \boldsymbol{c}), \mathit{next}(x, x'), \mathit{step}(t, t')\\
& \qquad \to \mathit{state}(t', \boldsymbol{s'}), \mathit{tape}(t', x, \boldsymbol{c'}), \mathit{pos}(t', x') \qquad \text{for } \delta(\boldsymbol{s}, \boldsymbol{c}) = \langle \boldsymbol{s'}, \boldsymbol{c'}, R\rangle\\
\rho_2 :\quad & \mathit{state}(t_0, \boldsymbol{q_0}), \mathit{state}(t, \boldsymbol{q_f}) \to \mathit{accept}()
\end{aligned}$$

A restricted chase C of the $\mathcal{I}$ and $\mathcal{R}(M, w)$ is a faithful simulation, irrespective of $\mathcal{I}$. (Even if $\mathcal{I} \neq \varnothing$ and for instance contains a cyclic *tape* or *step* predicate, this is no problem, as the initialization rule invents fresh nulls without connection to atoms in $\mathcal{I}$.) In case M holds and accepts w, the sequence $t = \rho_1\rho_2$ is a trail, witnessed by some such chase (representing the finite accepting run) and a corresponding step mapping. (The causal connection property is always met due to the $\mathit{state}(t_0, \boldsymbol{q_0})$ atoms occurring both in $\mathsf{head}(\rho_1)$ and $\mathsf{body}^+(\rho_2)$.) Otherwise, no chase with appropriate step mapping exists and t cannot be a trail. □

B Proof for Section 4

Lemma 5. *Every trail can be generalised by a decoupled chain.*

Proof. Let $t = \hat{\rho}_1 \ldots \hat{\rho}_n$ be a trail. By Definition 7, t is $\exists$-disjoint and there must be a witnessing chase C with step mapping $f : \{1, \ldots, n\} \to \mathsf{steps}_C$, such that $\mathsf{match}_C(f(x)) = \langle \hat{\rho}_x, \omega\rangle$ and $\mathsf{head}(\hat{\rho}_x) \cap \mathsf{body}^+(\hat{\rho}_{x+1}) \neq \varnothing$.

For each $i \in \{1, \ldots, n\}$, consider an injective replacement $\theta_1 = \mathrm{id}$ and for $i > 1$ $\theta_i : \mathsf{var}_\forall(\hat{\rho}_i) \setminus \mathsf{var}(\mathsf{head}(\hat{\rho}_{i-1})) \to \boldsymbol{V} \setminus (\bigcup_{j\in\{1,\ldots,i\}} \mathsf{var}(\hat{\rho}_j))$ of stale variables with fresh ones (as in Example 8). The sequence $\mathsf{decoupled}(t) = \hat{\rho}_1\theta_1, \ldots, \hat{\rho}_n\theta_n$ is clearly decoupled (and remains $\exists$-disjoint). It generalises t with $\sigma = \theta_1^{-1} \circ \ldots \circ \theta_n^{-1}$. (The individual replacements are invertible because they are injective. This function composition assumes that the domains of θ_i^{-1} are extended to $\boldsymbol{V}$ by identity in the usual way.)

To show that $\mathsf{decoupled}(t)$ is a chain, by Definition 10, we have to show for each $k \in \{1, \ldots, n-1\}$ that $\rho_{\hat{\rho}_1\theta_1\ldots\hat{\rho}_k\theta_k} \prec\!\!\!\prec^+ \hat{\rho}_{k+1}\theta_{k+1}$. We have $\hat{\rho}_i\theta_i\sigma = \hat{\rho}_i$ for all $i \in \{1, \ldots, n\}$, and hence also $\rho_{\hat{\rho}_1\theta_1\ldots\hat{\rho}_i\theta_i}\sigma = \rho_{\hat{\rho}_1\ldots\hat{\rho}_n}$ for the corresponding chain rules (see Equation (2)). So $\langle \rho_i\theta_i, \omega\rangle \in \mathsf{matches}_\mathcal{R}(\mathcal{I}\omega^{-1}\theta_i\omega)$ iff $\langle \rho_i, \omega\rangle \in \mathsf{matches}_\mathcal{R}(\mathcal{I})$ and $\langle \rho_{\hat{\rho}_1\theta_1\ldots\hat{\rho}_i\theta_i}, \omega\rangle \in \mathsf{matches}_\mathcal{R}(\bigcup_{j\in\{1,\ldots,i\}} \mathcal{I}\omega^{-1}\theta_j\omega)$ iff $\langle \rho_{\hat{\rho}_1\ldots\hat{\rho}_n}, \omega\rangle \in \mathsf{matches}_\mathcal{R}(\mathcal{I})$. Since $\mathcal{I}\omega^{-1}\theta_i\omega \subseteq \bigcup_{j\in\{1,\ldots,i\}} \mathcal{I}\omega^{-1}\theta_i\omega$ and none of

the negative body atoms of the matching rules can match in any of the copies, we further have that $\mathsf{matches}_{\mathcal{R}}(\mathcal{I}\omega^{-1}\theta_i\omega) \subseteq \mathsf{matches}_{\mathcal{R}}(\bigcup_{j\in\{1,\dots,i\}} \mathcal{I}\omega^{-1}\theta_j\omega)$.

Let $k \in \{1,\dots,n-1\}$ and define

$$\begin{aligned}\mathcal{I}_\delta &\coloneqq \left(\mathsf{head}(\rho_{\hat\rho_1\theta_1\dots\hat\rho_k\theta_k}) \setminus \mathsf{body}^+(\rho_{\hat\rho_1\theta_1\dots\hat\rho_k\theta_k})\right)\omega,\\ \mathcal{I}_b &\coloneqq \textstyle\bigcup_{j\in\{1,\dots,k\}} \mathsf{db}_C(f(k))\omega^{-1}\theta_j\omega, \text{ and}\\ \mathcal{I}_a &\coloneqq \mathcal{I}_b \setminus \mathcal{I}_\delta.\end{aligned}$$

Now consider the matches $\lambda_1 \coloneqq \langle \rho_{\hat\rho_1\theta_1\dots\hat\rho_k\theta_k}, \omega\rangle$ and $\lambda_2 \coloneqq \langle \hat\rho_{k+1}\theta_{k+1}, \omega\rangle$. We have $\lambda_1 \in \mathsf{matches}_{\mathcal{R}}(\bigcup_{j\in\{1,\dots,k\}} \mathsf{db}_C(f(k))\omega^{-1}\theta_j\omega) = \mathsf{matches}_{\mathcal{R}}(\mathcal{I}_b)$ (see above). Removal of $\mathcal{I}_\delta$ cannot have removed the match, so $\lambda_1 \in \mathsf{matches}_{\mathcal{R}}(\mathcal{I}_b \setminus \mathcal{I}_\delta) = \mathsf{matches}_{\mathcal{R}}(\mathcal{I}_a)$. The match λ_1 is unsatisfied over $\mathcal{I}_a$, as $\mathcal{I}_\delta$ is not contained in $\mathcal{I}_a$. Thus, $\lambda_1 \in \mathsf{unsat}_{\mathcal{R}}(\mathcal{I}_a)$ and the database $\mathcal{I}_b$ was obtained by applying λ_1 to $\mathcal{I}_a$.

Analogously, we have $\lambda_2 \in \mathsf{unsat}_{\mathcal{R}}(\mathsf{db}_C(f(k))\omega^{-1}\theta_k\omega)$, i.e. $\lambda_2 \in \mathsf{unsat}_{\mathcal{R}}(\mathcal{I}_b)$. Also, $\lambda_2 \notin \mathsf{matches}_{\mathcal{R}}(\mathcal{I}_a)$, as it requires facts from $\mathcal{I}_\delta$ that are only present in $\mathcal{I}_b$. Hence, $\lambda_2 \in \mathsf{unsat}_{\mathcal{R}}(\mathcal{I}_b) \setminus \mathsf{matches}_{\mathcal{R}}(\mathcal{I}_a)$. □

C Proofs for Section 5

Lemma 7. *Let $t = \hat\rho_1 \dots \hat\rho_n$ be a trail and C a witnessing chase for t with step mapping f. If there is a chase step $i < f(n)$, such that $\mathsf{rule}_C(i)$ has an alternative match over $\mathsf{db}_C(f(n))$ but not over $\mathsf{db}_C(f(n)-1)$, then $\hat\rho_1 \prec_c^{\square} \mathsf{rule}_C(i)$.*

Proof. To show $\hat\rho_1 \prec_c^{\square} \mathsf{rule}_C(i)$, by Definition 13 we need to give a decoupled chain c that starts with $\hat\rho_1$, such that $\rho_c \prec^{\square} \mathsf{rule}_C(i)$. Obtain $c \in \mathsf{chains}(\mathcal{R})$ generalising t with σ, by injectively renaming stale variables as in Lemma 5. By Definition 5, we thus need to give databases $\mathcal{I}_a \subseteq \mathcal{I}_b$ and mappings μ_1, μ_2, with $\langle \mathsf{rule}_C(i), \mu_2\rangle \in \mathsf{unsat}_{\mathcal{R}}(\mathcal{I}'_a)$ and $\mathcal{I}_a = \mathcal{I}'_a \cup \mathsf{head}(\mathsf{rule}_C(i))\mu_2$ and $\langle \rho_c, \mu_1\rangle \in \mathsf{unsat}_{\mathcal{R}}(\mathcal{I}_a)$ and $\mathcal{I}_b = \mathcal{I}'_b \cup \mathsf{head}(\rho_c)\mu_1$, such that there is alternative match for $\langle \mathsf{rule}_C(i), \mu_2\rangle$ over $\mathcal{I}_b$ but not over $\mathcal{I}'_b$. Take $\mathcal{I}'_a = \mathsf{db}_C(i-1)$ and $\mathcal{I}_a = \mathsf{db}_C(i)$ with $\mu_2 = \mathsf{hom}_C(i)$. We clearly have $\langle \mathsf{rule}_C(i), \mu_2\rangle \in \mathsf{unsat}_{\mathcal{R}}(\mathcal{I}'_a)$ and $\mathcal{I}_a = \mathcal{I}'_a \cup \mathsf{head}(\mathsf{rule}_C(i))\mu_2$ by Definition 1. Take $\mathcal{I}'_b = \mathsf{db}_C(f(n)-1)$ and $\mathcal{I}_b = \mathsf{db}_C(f(n))$. It is clear that $\mathcal{I}_a \subseteq \mathcal{I}_b$, since $i \le f(n)-1$. Recall that by Equation (2), $\mathsf{body}^+(\rho_c) = \left(\bigcup_{i\in\{1,\dots,n\}} \mathsf{body}^+(\hat\rho_i)\right) \cup \left(\bigcup_{i\in\{1,\dots,n-1\}} \mathsf{head}(\hat\rho_i)\right)$ and $\mathsf{head}(\rho_c) = \mathsf{head}(\hat\rho_n)$. We now have:

- $\mathsf{body}^+(\rho_c)\sigma\omega \subseteq \mathcal{I}'_b$ and $\mathcal{I}'_b \not\models \mathsf{head}(\rho_c)\sigma\omega_\forall$, i.e. $\langle \rho_c, \sigma\omega\rangle \in \mathsf{unsat}_{\mathcal{R}}(\mathcal{I}'_b)$, and
- $\mathcal{I}_b = \mathsf{db}_C(f(n)+1) = \mathsf{db}_C(f(n)) \cup \mathsf{head}(\mathsf{rule}_C(f(n)))\mathsf{hom}_C(f(n))$
  $= \mathcal{I}'_b \cup \mathsf{head}(\hat\rho_n)\omega = \mathcal{I}'_b \cup \mathsf{head}(\rho_c)\omega.$

By assumption, we have an alternative match for $\mathsf{rule}_C(i)$ over $\mathcal{I}_b$ but not $\mathcal{I}'_b$. □

Lemma 8. *Let $t = \hat\rho_1 \dots \hat\rho_n$ be a trail and C a witnessing chase for t with step mapping f. If there is a chase step $i < f(n)$ such that $\mathsf{match}_C(i)$ is a match for $\mathsf{rule}_C(i)$ over $\mathsf{db}_C(f(n)-1)$ but not over $\mathsf{db}_C(f(n))$, then $\hat\rho_1 \prec_c^{-} \mathsf{rule}_C(i)$.*

Proof. Analogous to the above proof of Lemma 7, we show by Definition 14 $\hat{\rho}_1 \prec_c^- \mathsf{rule}_C(i)$ via chain c generalising t with σ as in Lemma 5. By Definition 4 we need to give database $\mathcal{I}_a$ and mappings μ_1, μ_2 with $\langle \rho_c, \mu_1 \rangle \in \mathsf{unsat}_\mathcal{R}(\mathcal{I}_a)$ and $\mathcal{I}_b = \mathcal{I}_a \cup \mathsf{head}(\rho_c)\mu_1$, such that $\langle \mathsf{rule}_C(i), \mu_2 \rangle \in \mathsf{unsat}_\mathcal{R}(\mathcal{I}_a) \setminus \mathsf{unsat}_\mathcal{R}(\mathcal{I}_b)$. This is the case for $\mathcal{I}_a = \mathsf{db}_C(f(n)-1)$, $\mu_1 = \sigma\omega$ and $\mu_2 = \mathsf{hom}_C(i)$. □

D Proofs for Section 6

Lemma 10. *Let $c_1, c_2 \in \mathsf{chains}(\mathcal{R})$ with $\ell(c_1) = \ell(c_2)$. If $c_1\hat{\rho}_1 \in \mathsf{chains}(\mathcal{R})$ then there is $\hat{\rho}_2 \in \mathsf{instances}(\mathcal{R})$ (both $\hat{\rho}_1$ and $\hat{\rho}_2$ belong to the same $\rho \in \mathcal{R}$), such that*

$$\text{(i) } c_2\hat{\rho}_2 \in \mathsf{chains}(\mathcal{R}) \qquad \textit{and} \qquad \text{(ii) } \ell(c_1\hat{\rho}_1) = \ell(c_2\hat{\rho}_2)\ .$$

We will use some auxiliary notation:

- When we say formula set, we mean a set of formulae of the form $\exists \boldsymbol{v}.\psi$ for conjunction of atoms ψ. This is essentially a set of pieces.
- For databases $\mathcal{I}_1$ and $\mathcal{I}_2$ and formula set $\mathcal{F}$, we write $I_1 \equiv_\mathcal{F} I_2$, if for all formulae $\exists \boldsymbol{v}.\tilde{\psi} \in \mathcal{F}$, we have that $\mathcal{I}_1 \models \exists \boldsymbol{v}.\tilde{\psi}\omega_\forall$ iff $\mathcal{I}_2 \models \exists \boldsymbol{v}.\tilde{\psi}\omega_\forall$.
- For database $\mathcal{I}$, we write $\mathsf{symbols}(\mathcal{I})$ for the set of constants and nulls occurring in facts of $\mathcal{I}$.
  For formula set $\mathcal{F}$, we write $\mathsf{symbols}(\mathcal{F}) = \bigcup_{\exists \boldsymbol{v}.\psi \in \mathcal{F}} \mathsf{var}_\forall(\exists \boldsymbol{v}.\psi)\omega$ for the set of constants and nulls injectively assigned to universal variables of $\mathcal{F}$ by ω.
- We will apply $\mathsf{variants}(\cdot)$ to formula sets, defined exactly like $\mathsf{variants}(\mathcal{R})$ (page 12) but using $\mathcal{F}$ instead of $\mathcal{R}$.

Proof. Let c_1 and c_2 be two decoupled chains with the same label. Then $\ell^\mathsf{C}(c_1) = \ell^\mathsf{C}(c_2)$, $\ell^\mathsf{S}(c_1) = \ell^\mathsf{S}(c_2)$, and $\ell^\mathsf{N}(c_1) = \ell^\mathsf{N}(c_2)$. Let $\hat{\rho}_n$ and $\hat{\rho}_m$ be the last rule instances of c_1 and c_2, respectively. Further, let θ_n and θ_m be the two (unique) injective mappings, such that $\hat{\rho}_n = \mathsf{orig}(\hat{\rho}_n)\theta_n$ and $\hat{\rho}_m = \mathsf{orig}(\hat{\rho}_m)\theta_m$.

Let $\hat{\rho}_1 \in \mathsf{instances}(\mathcal{R})$, such that $c_1\hat{\rho}_1$ is a decoupled chain. By Definition 11, $c_1\hat{\rho}_1$ is decoupled if and only if all variables from $\mathsf{var}(\hat{\rho}_1) \setminus \mathsf{var}(\mathsf{head}(\hat{\rho}_n))$ are fresh w.r.t. c_1. By Definitions 10, $c_1\hat{\rho}_1$ is a chain if and only if $\rho_{c_1} \lessdot^+ \hat{\rho}_1$. Since θ_n is injective, we can apply its inverse on both sides and get $\rho_{c_1}\theta_n^{-1} \lessdot^+ \hat{\rho}_1\theta_n^{-1}$, which means by Definition 8 that, for the database $\mathcal{I}_1 \coloneqq (\mathsf{body}^+(\rho_{c_1}) \cup \mathsf{head}(\rho_{c_1}) \cup \mathsf{body}^+(\hat{\rho}_1))\theta_n^{-1}\omega$, we have

(1.a) the intersection between the atom sets $(\mathsf{head}(\rho_{c_1}) \setminus \mathsf{body}^+(\rho_{c_1}))\theta_n^{-1}$ and $\mathsf{body}^+(\hat{\rho}_1)\theta_n^{-1}$ is non-empty, and
(1.b) the database $\mathcal{I}_1$ does not satisfy $\exists \boldsymbol{v}.\, \mathsf{head}(\hat{\rho}_1)\theta_n^{-1}\omega_\forall$, and
(1.c) none of the negative body atoms of $\hat{\rho}_1\theta_n^{-1}$ are in $\mathcal{I}_1$.

Let θ_1 be the (unique) injective mapping, such that $\hat{\rho}_1 = \mathsf{orig}(\hat{\rho}_1)\theta_1$.

To prove (i), we want to construct an instance $\hat{\rho}_2$, such that $c_2\hat{\rho}_2$ is a chain, which is the case if and only if $\rho_{c_2} \lessdot^+ \hat{\rho}_2$. Let $\hat{\rho}_2 \coloneqq \hat{\rho}_1\theta_n^{-1}\theta_m$ be that instance. Since θ_m is injective, we can apply its inverse on both sides and get $\rho_{c_2}\theta_m^{-1} \lessdot^+ \hat{\rho}_2\theta_m^{-1}$. Unfolding $\hat{\rho}_2$, the right-hand side becomes $\hat{\rho}_2\theta_m^{-1} = \hat{\rho}_1\theta_n^{-1}\theta_m\theta_m^{-1} = \hat{\rho}_1\theta_n^{-1}$. Unpacking Definition 8 for $\rho_{c_2}\theta_m^{-1} \lessdot^+ \hat{\rho}_1\theta_n^{-1}$, we need to show for database $\mathcal{I}_2 \coloneqq (\mathsf{body}^+(\rho_{c_2}) \cup \mathsf{head}(\rho_{c_2}) \cup \mathsf{body}^+(\hat{\rho}_2))\theta_m^{-1}\omega$, that

(2.a) the intersection between the atom sets $(\mathsf{head}(\rho_{c_2}) \setminus \mathsf{body}^+(\rho_{c_2}))\theta_m^{-1}$ and $\mathsf{body}^+(\hat{\rho}_1)\theta_n^{-1}$ is non-empty, and
(2.b) the database $\mathcal{I}_2$ does not satisfy $\exists \boldsymbol{v}.\, \mathsf{head}(\hat{\rho}_1)\theta_n^{-1}\omega_\forall$, and
(2.c) none of the negative body atoms of $\hat{\rho}_1\theta_n^{-1}$ are in $\mathcal{I}_2$.

First, we derive (2.a): Recall that $\ell^{\mathsf{C}}(c_1) = \ell^{\mathsf{C}}(c_2)$ means by Equation (4), that the pieces of $\mathsf{orig}(\hat{\rho}_n)$, which are not contained in $\mathsf{body}^+(\rho_{c_1})$ under θ_n, are exactly the pieces of $\mathsf{orig}(\hat{\rho}_m)$, which are not contained in $\mathsf{body}^+(\rho_{c_2})$ under θ_m. Also recall that $\mathsf{head}(\rho_{c_1}) = \mathsf{head}(\mathsf{orig}(\hat{\rho}_n))\theta_n$ and $\mathsf{head}(\rho_{c_2}) = \mathsf{head}(\mathsf{orig}(\hat{\rho}_m))\theta_m$. Therefore, $(\mathsf{head}(\rho_{c_1}) \setminus \mathsf{body}^+(\rho_{c_1}))\theta_n^{-1} = (\mathsf{head}(\rho_{c_2}) \setminus \mathsf{body}^+(\rho_{c_2}))\theta_m^{-1}$. With this equality, (2.a) directly follows from (1.a).

Next, we derive (2.b): Expanding Equation (5) at $\ell^{\mathsf{S}}(c_1) = \ell^{\mathsf{S}}(c_2)$, we get that $(\mathsf{body}^+(\rho_{c_1}) \cup \mathsf{head}(\rho_{c_1}))\omega \models \exists \boldsymbol{v}.\, \tilde{\psi}(\theta_n\omega)_\forall$ if and only if $(\mathsf{body}^+(\rho_{c_2}) \cup \mathsf{head}(\rho_{c_2}))\omega \models \exists \boldsymbol{v}.\, \tilde{\psi}(\theta_m\omega)_\forall$ for all $\exists \boldsymbol{v}.\, \tilde{\psi} \in \mathcal{F} \subseteq \mathsf{parts}(\mathsf{pieces}(\mathsf{variants}(\mathcal{R})))$. As θ_n and θ_m are both injective, we can apply their inverses to learn that $\mathcal{I}_1 \models \exists \boldsymbol{v}.\, \tilde{\psi}\omega_\forall$ if and only if $\mathcal{I}_2 \models \exists \boldsymbol{v}.\, \tilde{\psi}\omega_\forall$. Using the forward direction of this equivalence, (2.b) follows from (1.b). The defining condition for $\equiv_{\mathcal{F}}$ is exactly this equivalence, so $\mathcal{I}_1 \equiv_{\mathcal{F}} \mathcal{I}_2$, which will later be needed to argue for (ii).

Finally, we derive (2.c): By Equation (6), $\ell^{\mathsf{N}}(c_1) = \ell^{\mathsf{N}}(c_2)$ says that a variant $\check{\rho}$ has a negative body atom falling into $\mathcal{I}_1$ under θ_n if and only if it has a negative body atom falling into $\mathcal{I}_2$ under θ_m. By (1.c), the variant $\mathsf{orig}(\hat{\rho}_1) = \hat{\rho}_1\theta_n^{-1}$ cannot be one of them, so (2.c) follows.

Having (2.a)–(2.c), we conclude that $c_2\hat{\rho}_2$ is a chain. Note that $c_2\hat{\rho}_2$ might not be decoupled, if the variables from $\mathsf{var}(\hat{\rho}_1) \setminus \mathsf{var}(\mathsf{head}(\hat{\rho}_n))$, which necessarily are fresh w.r.t. c_1, are not also fresh w.r.t. c_2. If so, just take any injective mapping $\theta' : \mathsf{stale}(c_2\hat{\rho}_2) \to \boldsymbol{V} \setminus \bigcup_{i\in\{1,\dots,m\}} \mathsf{var}(\hat{\rho}_i)$, and $c_2\hat{\rho}_2\theta'$ is clearly decoupled. As θ' acts only on the variables in $\hat{\rho}_2$ not shared with the head of c_2's final instance, $\rho_{c_2} \prec^+ \hat{\rho}_2\theta'$ holds iff $\rho_{c_2} \prec^+ \hat{\rho}_2$ (by an argument similar to the proof of Lemma 5), i.e. $c_2\hat{\rho}_2\theta'$ is a (decoupled) chain.

To prove (ii), we first establish:

Claim (‡). Let $\mathcal{F}$ be a formula set, such that $\mathsf{parts}(\mathsf{variants}(\mathcal{F})) = \mathcal{F}$. Let $\mathcal{I}_1$, $\mathcal{I}_2$, and X be databases, such that $(\mathsf{symbols}(\mathcal{I}_1 \cup \mathcal{I}_2)) \cap \mathsf{symbols}(X) \subseteq \mathsf{symbols}(\mathcal{F})$. Then $\mathcal{I}_1 \equiv_{\mathcal{F}} \mathcal{I}_2$ implies that $\mathcal{I}_1 \cup X \equiv_{\mathcal{F}} \mathcal{I}_2 \cup X$.

Proof. Let databases $\mathcal{I}_1 \equiv_{\mathcal{F}} \mathcal{I}_2$, and X as well as formula set $\mathcal{F}$ be as in the claim. This means that $\mathcal{I}_1$ and $\mathcal{I}_2$ agree on satisfaction of formulae from $\mathcal{F}$ under $\omega_\forall$ and that X uses only fresh symbols, except for $\mathsf{symbols}(F)$. Let $\exists \boldsymbol{v}.\tilde{\psi}$ be any formula from $\mathcal{F}$.

To prove the claim, we need to show that either both $\mathcal{I}_1 \cup X \models \exists \boldsymbol{v}.\tilde{\psi}\omega_\forall$ and $\mathcal{I}_2 \cup X \models \exists \boldsymbol{v}.\tilde{\psi}\omega_\forall$ or neither $\mathcal{I}_1 \cup X \not\models \exists \boldsymbol{v}.\tilde{\psi}\omega_\forall$ nor $\mathcal{I}_2 \cup X \not\models \exists \boldsymbol{v}.\tilde{\psi}\omega_\forall$.

In case that $\mathcal{I}_1 \models \exists \boldsymbol{v}.\tilde{\psi}\omega_\forall$, then $\mathcal{I}_2$ must also satisfy it because of $\mathcal{I}_1 \equiv_{\mathcal{F}} \mathcal{I}_2$. As satisfaction for pieces is monotone, we directly get that $\mathcal{I}_1 \cup X \models \exists \boldsymbol{v}.\tilde{\psi}\omega_\forall$ and $\mathcal{I}_2 \cup X \models \exists \boldsymbol{v}.\tilde{\psi}\omega_\forall$.

Consider the case that $\mathcal{I}_1 \not\models \exists \boldsymbol{v}.\tilde{\psi}\omega_\forall$. By $\mathcal{I}_1 \equiv_{\mathcal{F}} \mathcal{I}_2$, we must also have $\mathcal{I}_2 \not\models \exists \boldsymbol{v}.\tilde{\psi}\omega_\forall$. Suppose that $\mathcal{I}_2 \cup X \models \exists \boldsymbol{v}.\tilde{\psi}\omega_\forall$. So there exists a mapping $\mu_\exists : \boldsymbol{v} \to \boldsymbol{C} \cup \boldsymbol{N}$ such that $\tilde{\psi}\omega_\forall\mu_\exists \subseteq \mathcal{I}_2 \cup X$. It remains to show that $\mathcal{I}_1 \cup X \models \exists \boldsymbol{v}.\tilde{\psi}\omega_\forall$.

The image of $\mu_\exists$ can be separated into three disjoint sets $A = \mathsf{symbols}(\mathcal{F})$ and $B = \mathsf{symbols}(X) \setminus A$ and finally $C = \mathsf{symbols}(\mathcal{I}_2) \setminus A$. Accordingly, we can split $\boldsymbol{v}$ into three disjoint sets $\boldsymbol{v}_A \dot\cup \boldsymbol{v}_B \dot\cup \boldsymbol{v}_C$ where $\boldsymbol{v}_S = \{v \in \boldsymbol{v} \mid \mu_\exists(v) \in S\}$ for a set S. And finally, we can split $\mu_\exists$ into three separate mappings $\mu_\exists^A : \boldsymbol{v}_A \to A$, $\mu_\exists^B : \boldsymbol{v}_B \to B$, and $\mu_\exists^C : \boldsymbol{v}_C \to C$, such that $\mu_\exists = \mu_\exists^A \circ \mu_\exists^B \circ \mu_\exists^C$.[8]

As $\tilde\psi\omega_\forall\mu_\exists \subseteq \mathcal{I}_2 \cup X$, we can write $\tilde\psi$ as the disjoint union of two sets $\tilde\psi_1$ and $\tilde\psi_2$, such that $\tilde\psi_1\omega_\forall\mu_\exists \subseteq \mathcal{I}_2$ and $\tilde\psi_2\omega_\forall\mu_\exists \subseteq X$. Clearly, the image of existentials from the first set is $\mu_\exists[\mathsf{var}_\exists(\tilde\psi_1)] \subseteq \mathsf{symbols}(\mathcal{I}_2) = A \cup C$ and the image of existentials from the second set is $\mu_\exists[\mathsf{var}_\exists(\tilde\psi_2)] \subseteq \mathsf{symbols}(X) = A \cup B$. So we have $\tilde\psi_1\omega_\forall\mu_\exists^A\mu_\exists^C \subseteq \mathcal{I}_2$ and $\tilde\psi_2\omega_\forall\mu_\exists^A\mu_\exists^B \subseteq X$.

Let $\check\rho$ be a variant with $\exists\boldsymbol{v}.\,\tilde\psi \in \mathsf{parts}(\check\rho)$. Applying the variable mapping $(\mu_\exists^A\omega^{-1}) : \mathsf{var}(\mathcal{R}) \to \mathsf{var}(\mathcal{R})$ yields the variant $\check\rho(\mu_\exists^A\omega^{-1})$. The pieces of the formula $\exists\boldsymbol{v}_C.\,\tilde\psi_1(\mu_\exists^A\omega^{-1})$ are subformulae of $\check\rho(\mu_\exists^A\omega^{-1}) \in \mathsf{variants}(\mathcal{F})$. Since each of them is satisfied by $\mathcal{I}_2$ (with $\mu_\exists^C$) they must also be satisfied by $\mathcal{I}_1$, i.e. we must be able to find a $\mu_\exists^{C'} : \boldsymbol{v}_C \to \mathsf{symbols}(\mathcal{I}_1)$, such that $\tilde\psi_1(\mu_\exists^A\omega^{-1})\omega_\forall\mu_\exists^{C'} = \tilde\psi_1\omega_\forall\mu_\exists^A\mu_\exists^{C'} \subseteq \mathcal{I}_1$. Thus, $\mathcal{I}_1 \cup X \models \exists\boldsymbol{v}.\,\tilde\psi\omega_\forall$ with $\mu_\exists^A\mu_\exists^B\mu_\exists^{C'}$. □

For (ii), we need to show that $\ell(c_1\hat\rho_1) = \ell(c_2\hat\rho_2)$. We will show this for the three components of the label separately.

Clearly, we have $\ell^{\mathsf{C}}(c_1\hat\rho_1) = \ell^{\mathsf{C}}(c_2\hat\rho_2)$, because the final instances $\hat\rho_1$ and $\hat\rho_2$ both correspond to the same variant $\mathsf{orig}(\hat\rho_1)$. In (1.a) and (2.a) we thus identified the same non-empty intersection between the body of this variant and the heads of the chain rules minus their bodies.

As noted, $\ell^{\mathsf{S}}(c_1) = \ell^{\mathsf{S}}(c_2)$ gives $\mathcal{I}_1 \equiv_{\mathcal{F}} \mathcal{I}_2$. Since $\mathsf{parts}(\mathsf{variants}(\cdot))$ is idempotent and the set $\mathcal{F}$ used for computing ℓ^{S} is $\mathsf{parts}(\mathcal{H})$ (or with $\mathcal{R}_D$-closure $\mathsf{parts}(\mathcal{H} \cup \mathcal{B})$), and $\mathcal{H}$ and $\mathcal{B}$ are closed under $\mathsf{variants}(\cdot)$, it fulfils the requirement of the claim ‡. The variables used universally in $\mathcal{F}$ are from $\mathsf{var}(\mathcal{R})$, i.e. X may share only symbols $\mathsf{var}(\mathcal{R})\omega$ with the union of $\mathcal{I}_1$ and $\mathcal{I}_2$. Note that $\mathsf{symbols}(\mathcal{I}_1 \cup \mathcal{I}_2) = (\mathsf{symbols}(\mathcal{I}_1) \cup \mathsf{symbols}(\mathcal{I}_2))$. The set of facts $X = (\mathsf{body}^+(\mathsf{orig}(\hat\rho_1)) \cup \mathsf{head}(\mathsf{orig}(\hat\rho_1)))\omega$ meets the conditions for claim ‡, since the instance is decoupled w.r.t. both chains c_1 and c_2. So we find that $\mathcal{I}_1 \cup X \equiv_{\mathcal{F}} \mathcal{I}_2 \cup X$ by (‡). This set X is exactly the set of facts that is added to the respective databases for computing $\ell^{\mathsf{S}}(c_1)$ and $\ell^{\mathsf{S}}(c_2)$.

Lastly, we clearly have $\ell^{\mathsf{N}}(c_1\hat\rho_1) = \ell^{\mathsf{N}}(c_2\hat\rho_2)$, as the fact set X which adds to the databases are again identical. □

Theorem 11. *The language $\mathcal{L}_\mathcal{R}$ is regular.*

Claim (†). Let $w_1, w_2 \in \mathcal{L}_\mathcal{R}$ be two words with the same final letter. For any $v \in \Sigma^*$, $w_1 v \in \mathcal{L}_\mathcal{R}$ iff $w_2 v \in \mathcal{L}_\mathcal{R}$.

[8] For a function $f : A \to B$, we call $f^{\mathrm{id}} : A \cup B \to A \cup B$ with $x \mapsto f(x)$ for $x \in A$ and $x \mapsto x$ for $x \in B \setminus A$ the identity extension of f. Given two functions $f : A_1 \to B$ and $g : A_2 \to B$ where $A_1 \cap A_2 = \varnothing$, we write $f \circ g : A_1 \dot\cup A_2 \to B$ for the composition $f^{\mathrm{id}} \circ g^{\mathrm{id}}$ of their identify extensions.

Proof (by induction on $|v|$). For the induction base, the length of v is zero, i.e. $v = \epsilon$ (the empty word) and $w_2 v = w_2$. By assumption $w_2 \in \mathcal{L}_\mathcal{R}$ and so directly $w_2 v \in \mathcal{L}_\mathcal{R}$.

The induction hypothesis (IH) is that for any word $v' \in \Sigma^*$ and symbol $c \in \Sigma$, $w_1 v' \in \mathcal{L}_\mathcal{R} \iff w_2 v' \in \mathcal{L}_\mathcal{R}$ implies $w_1 v' c \in \mathcal{L}_\mathcal{R} \iff w_2 v' c \in \mathcal{L}_\mathcal{R}$.

For the induction step, let $v = v'c$ for $v' \in \Sigma^*$ and $c \in \Sigma$. If $v' = \epsilon$, the final letter of $w_1 v'$ is the final letter of w_1 and the final letter of $w_2 v'$ is the final letter of w_2, which are the same by assumption. Otherwise, the final letter of both $w_1 v'$ and $w_2 v'$ is the final letter of v'. Therefore, $w_1 v'$ and $w_2 v'$ have identical final letters in both cases and hence correspond to equi-labelled chains. Applying Lemma 10 finishes the induction. □

- For sets S and T, an equivalence relation $\sim$ over S distinguishes elements based on a predicate $p : S \to T$, if $x \sim y$ iff $p(x) = p(y)$ for all $x, y \in S$.
- For a language $\mathcal{L} \subseteq \Sigma^*$, the extensions of a word $w \in \mathcal{L}$ is the set of all $v \in \Sigma^*$ for which $wv \in \mathcal{L}$.
- By Myhill-Nerode, a language is regular, if it is possible to give an equivalence relation over its words that has finitely many equivalence classes and distinguishes words based on their extensions.

Proof (of Theorem 11). We define an equivalence relation over words in the language $\mathcal{L}_\mathcal{R}$ based on their final letters

$$\sim_{\mathcal{L}_\mathcal{R}} \coloneqq \{\langle w(c_1), w(c_2)\rangle \mid c_1, c_2 \in \mathsf{chains}(\mathcal{R}), \ell(c_1) = \ell(c_2)\}. \tag{14}$$

Clearly, $\sim_{\mathcal{L}_\mathcal{R}}$ has at most $|\Sigma|$-many equivalence classes, which is finitely many as the alphabet is finite. By (†), two words from $\mathcal{L}_\mathcal{R}$ that share the same label ($\hat{=}$ final letter), i.e. are $\sim_{\mathcal{L}_\mathcal{R}}$-equivalent, have the same extensions.

Hence, $\sim_{\mathcal{L}_\mathcal{R}} \cup (\Sigma^* \setminus \mathcal{L}_\mathcal{R})^2$ meets the conditions of Myhill-Nerode, which makes $\mathcal{L}_\mathcal{R}$ a regular language. □

Claim. For $c_1, c_2 \in \mathsf{chains}(\mathcal{R})$ with $\ell(c_1) = \ell(c_2)$, we have $\rho_{c_1} \prec^\square \rho$ iff $\rho_{c_2} \prec^\square \rho$ and $\rho_{c_1} \prec^- \rho$ iff $\rho_{c_2} \prec^- \rho$.

Proof. To compute $\rho_{c_i} \prec^\square \rho$ (see Definition 5) and $\rho_{c_i} \prec^- \rho$ (see Definition 4) for $i \in \{1, 2\}$, it suffices to examine unifiable subsets of $\mathsf{head}(\rho)$ or $\mathsf{body}^-(\rho)$ with $\mathsf{head}(\rho_{c_i})$ (c.f. efficient algorithms of [9]). Remaining checks concern the satisfaction of the heads of the involved rules on representative databases. Let $\hat{\rho}_{n_i}$ be the final instances in c_i, with unique θ_{n_i}, such that $\hat{\rho}_{n_i} = \mathsf{orig}(\hat{\rho}_{n_i})\theta_{n_i}$. Again, we can equivalently test for $\rho_{c_i}\theta_{n_i}^{-1} \prec^\square \rho\theta_{n_i}^{-1}$ and $\rho_{c_i}\theta_{n_i}^{-1} \prec^- \rho\theta_{n_i}^{-1}$ (as θ_{n_i} is revertible and $\theta_{n_i}^{-1}$ is injective). These will involve $(\mathsf{body}^+(\rho_{c_i}) \cup \mathsf{head}(\rho_{c_i}))\theta_{n_i}^{-1}\omega$, and we stated before that these are $\equiv_\mathcal{F}$-equivalent due to $\ell^\mathsf{S}(c_1) = \ell^\mathsf{S}(c_2)$ (see proof of Lemma 10 (i)) and satisfy the same rule heads. □

E Details on Introductory Example

We will now revisit the introductory examples (cf. Section 1, Problems 1 and 2). Here, we refer to the N3 rule in line i by ρ_i. Figure 1 depicts a graphical representation of the reliance relations between these rules. Evidently, this yields an acyclic graph for L8–L11 on the right, which means that $\mathcal{R}_{\mathrm{P2}} = \{\rho_8, \rho_9, \rho_{10}, \rho_{11}\}$ is *fully stratified* (and thus also *chain-stratified*). The graph on the left, however, has cycles through negative reliances $\rho_1 \prec^- \rho_2$ and $\rho_4 \prec^- \rho_2$, i.e. $\mathcal{R}_{\mathrm{P1}} = \{\rho_1, \rho_2, \rho_3, \rho_4\}$ and $\mathcal{R}'_{\mathrm{P1}} = \mathcal{R}_{\mathrm{P1}} \cup \{\rho_5, \rho_6, \rho_7\}$ are not *fully stratified*. Below, we will examine $\mathcal{R}_{\mathrm{P1}}$ and $\mathcal{R}'_{\mathrm{P1}}$ in detail, to see that — while neither is *chain-stratified* either (see Example 15) — $\mathcal{R}'_{\mathrm{P1}}$ is *chain-stratified under constraints* (as defined in Section 7).

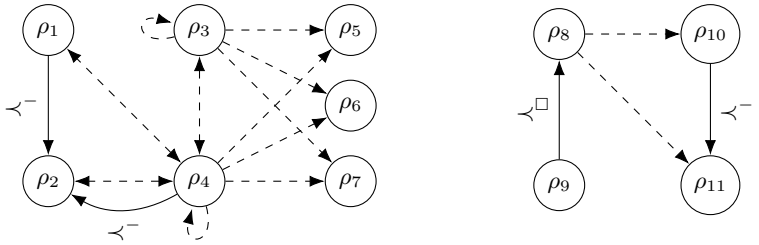


Figure 1. Reliance graph of L1–L7 (left) and L8–L11 (right); dashed lines are $\prec^+$

Let us first rewrite L1–L7 in more compact syntax and using fewer variables.

$$\begin{aligned}
\rho_1 &: \quad t(?x, pa, ?y) \to t(?y, ty, St) \\
\rho_2 &: \quad t(?x, te, ?y), \neg t(?y, ty, St) \to t(?x, ex, ?y) \\
\rho_3 &: \quad t(?p, spo, ?y), t(?y, spo, ?q) \to t(?p, spo, ?q) \\
\rho_4 &: \quad t(?p, spo, ?q), t(?x, ?p, ?y) \to t(?x, ?q, ?y)
\end{aligned}
\qquad
\begin{aligned}
\rho_5 &: \quad t(ex, spo, pa) \to \bot \\
\rho_6 &: \quad t(ex, spo, spo) \to \bot \\
\rho_7 &: \quad t(ex, spo, ty) \to \bot
\end{aligned}$$

There are cycles through negative reliances, e.g. $\rho_2 \prec^+ \rho_4 \prec^+ \rho_1 \prec^- \rho_2$ or just $\rho_2 \prec^+ \rho_4 \prec^- \rho_2$. All $\prec^-$-edges end in ρ_2, so we only need to examine chains starting with instances of ρ_2. As explained earlier, the ruleset is indeed not *chain stratified*, but it is *chain stratified under constraints* due to the constraints ρ_5, ρ_6 and ρ_7. Algorithm 1 will consider the following chains in the given order:

ρ_2, ρ_4 : $t(?x, te, ?y), t(?x, ex, ?y), t(ex, spo, ?q_1) \to t(?x, ?q_1, ?y)$
- chain accepted
- $\prec^- \rho_2$ rejected because it would require $?q_1 \mapsto ty$, $?y \mapsto St$, violating constraint ρ_7

ρ_2, ρ_4, ρ_1 : $t(?x, te, ?y), t(?x, ex, ?y), t(ex, spo, pa), t(?x, pa, ?y) \to t(?y, ty, St)$
- chain rejected because of constraint ρ_5

ρ_2, ρ_4, ρ_2 : $t(?x, te, ?y), t(?x, ex, ?y), t(ex, spo, te), t(?x, te, ?y) \to t(?x, ex, ?y)$
- chain rejected because the head is already entailed

ρ_2, ρ_4, ρ_3 : $t(?x, te, ?y), t(?x, ex, ?y), t(ex, spo, spo), t(?x, spo, ?y), t(?y, spo, ?z) \rightarrow t(?x, spo, ?z)$
- chain rejected because of constraint ρ_6

ρ_2, ρ_4, ρ_4 :
$[?p/?q_1, ?q/?q_2]$: $t(?x, te, ?y), t(?x, ex, ?y), t(ex, spo, ?q_1), t(?x, ?q_1, ?y), t(?q_1, spo, ?q_2) \rightarrow t(?x, ?q_2, ?y)$
- chain accepted
- $\prec^-$ ρ_2 rejected because it would require $?q_2 \mapsto ty$, $?y \mapsto St$, such that ρ_3 derives $t(ex, spo, ty)$, violating constraint ρ_7

$[?q_1/spo, ?p/?x, ?q/?y]$: $t(?x, te, ?y), t(?x, ex, ?y), t(ex, spo, spo), t(?x, spo, ?y), t(?x_1, ?x, ?y_1) \rightarrow t(?x_1, ?y, ?y_1)$
- chain rejected because of constraint ρ_6

$\rho_2, \rho_4, \rho_4, \rho_1$: $t(?x, te, ?y), t(?x, ex, ?y), t(ex, spo, ?q_1), t(?x, ?q_1, ?y), t(?q_1, spo, pa), t(?x, pa, ?y) \rightarrow t(?y, ty, St)$
- chain rejected because ρ_3 derives $t(ex, spo, pa)$, violating constraint ρ_5

$\rho_2, \rho_4, \rho_4, \rho_2$: $t(?x, te, ?y), t(?x, ex, ?y), t(ex, spo, ?q_1), t(?x, ?q_1, ?y), t(?q_1, spo, te), t(?x, te, ?y) \rightarrow t(?x, ex, ?y)$
- chain rejected because the head is already entailed

$\rho_2, \rho_4, \rho_4, \rho_3$: $t(?x, te, ?y), t(?x, ex, ?y), t(ex, spo, ?q_1), t(?x, ?q_1, ?y), t(?q_1, spo, spo), t(?x, spo, ?y), t(?y, spo, ?z) \rightarrow t(?x, spo, ?z)$
- chain rejected because ρ_3 derives $t(ex, spo, spo)$, violating constraint ρ_6

$\rho_2, \rho_4, \rho_4, \rho_4$:
$[?p/?q_2, ?q/?q_3]$: $t(?x, te, ?y), t(?x, ex, ?y), t(ex, spo, ?q_1), t(?x, ?q_1, ?y), t(?q_1, spo, ?q_2), t(?x, ?q_2, ?y), t(?q_2, spo, ?q_3) \rightarrow t(?x, ?q_3, ?y)$
- chain accepted (...but continuing like this will repeat label)
- $\prec^-$ ρ_2 rejected because it would require $?q_3 \mapsto ty$, $?y \mapsto St$, such that two ρ_3 applications derive $t(ex, spo, ty)$, violating constraint ρ_7

$[?q_2/spo, ?p/?x, ?q/?y]$: $t(?x, te, ?y), t(?x, ex, ?y), t(ex, spo, ?q_1), t(?x, ?q_1, ?y), t(?q_1, spo, spo), t(?x, spo, ?y), t(?x_1, ?x, ?y_1) \rightarrow t(?x_1, ?y, ?y_1)$
- chain rejected because ρ_3 derives $t(ex, spo, spo)$, violating constraint ρ_6

None of these passed the conditions in A14 or A16 of the algorithm, so no new edges were added to $\prec$. Therefore, A6 will not find cycles and A17 is reached, concluding that $\mathcal{R}'_{\mathrm{P1}}$ indeed is *chain-stratified under constraints.*