No Rule is Forever: Datalog Reasoning with Rule Amendments (proof appendix)

Weiqin Xu
LIGM Univ Gustave Eiffel, CNRS
Marne la Vallée, France
weiqin.xu@univ-eiffel.fr

Riccardo Tommasini
LIRIS, INSA
Lyon, France
riccardo.tommasini@insa-lyon.fr

Olivier Curé
LIGM Univ Gustave Eiffel, CNRS
Marne la Vallée, France
olivier.cure@univ-eiffel.fr

References

[1] S. Abiteboul, R. Hull, and V. Vianu. *Foundations of Databases*. Addison-Wesley, 1995.

Appendix

This appendix provides detailed proofs of the various lemmas of this article. For the reader's convenience, the lemmas are restated. We begin by providing some additional definitions and results adapted from [1].

A. Additional definitions and results

For a semi-positive Datalog program P and an extensional database EDB over $\text{edb}(P)$, we define $P(\text{EDB})$ as the set of facts derivable from EDB using the rules of P. This semantics is defined as the smallest fixed point of the direct consequence relation T_P. For any instance K, $T_P(K)$ contains the facts present in K for predicates in $\text{edb}(P)$, as well as the facts that can be derived from P using the facts in K. For an extensional database EDB over $\text{edb}(P)$, $P(\text{EDB})$ corresponds to the limit of the increasing sequence $(T_P^k(\text{EDB}))_{k\geq 0}$.

For convenience, we extend the definition of the semantics of P to any instance I by taking $P(I)$ to be $P(I \mid \text{edb}(P))$, where $I \mid \text{edb}(P)$ denotes the restriction of the instance I to the relations in $\text{edb}(P)$.

The following lemma recalls some basic properties of the direct consequence relation T_P:

Lemma A.1. *Let P and P' be two semi-positive Datalog programs such that $P' \subseteq P$ and $\text{edb}(P') \subseteq \text{edb}(P)$.*

1) *For any instance I, $T_P(I)|\text{edb}(P) = I|\text{edb}(P)$.*
2) *For any instance I over $\text{edb}(P)$, $P(I)|\text{edb}(P) = I$.*
3) *For two instances $J' \subseteq J$ such that $J|\text{edb}(P') = J'|\text{edb}(P')$, we have $T_{P'}(J') \subseteq T_P(J)$.*

The following lemma relates the semantics of a program to that of its subprograms.

Lemma A.2. *Let $P' \subseteq P$ be semi-positive Datalog programs and let EDB be an extensional database over $\text{edb}(P)$. We have*

$$P'(\text{EDB}) \subseteq P(\text{EDB}).$$

Proof. We prove by induction that for all $k \geq 0$, $T_{P'}^k(\text{EDB}) \subseteq T_P^k(\text{EDB})$. The base case is immediate. For the induction step, assume that $T_{P'}^k(\text{EDB}) \subseteq T_P^k(\text{EDB})$. Using Lemma A.1.1, we can easily show that

$$\begin{aligned} & T_{P'}^k(\text{EDB}) \mid \text{edb}(P') \\ = \; & T_P^k(\text{EDB}) \mid \text{edb}(P') \\ = \; & \text{EDB} \mid \text{edb}(P'). \end{aligned}$$

Applying Lemma A.1.3, we obtain $T_{P'}^{k+1}(\text{EDB}) \subseteq T_P^{k+1}(\text{EDB})$. It follows that $P'(\text{EDB})$, which is the limit of the sequence $(T_{P'}^k(\text{EDB}))_{k\geq 0}$, is included in $P(\text{EDB})$, which is the limit of the sequence $(T_P^k(\text{EDB}))_{k\geq 0}$. □

B. Proof of Lemma 1

Lemma 1. *A Hyper Rule Dependency Graph will always be a Directed Acyclic Graph (DAG).*

Proof. We define the expression $a \mapsto b$, which represents that there exists a path from a to b in a directed graph. Assuming an $HRDG$ is not a DAG, then there must exist a limited number of hypernodes $h1, h2, .., hk \in H$ $(k > 1)$, where $h1 \mapsto h2 \mapsto ... \mapsto hk \mapsto h1$. According to the definition of $HRDG$, considering the positive case, if there exists a positive dependency between $h1$ and $h2$, then there must be two rules $r_{12} \in h1$ and $r_{21} \in h2$ and $(r_{12}, r_{21}) \in P$ (P represents the positive RDG), where $r_{12} \mapsto r_{21}$. Similarly, if $h2 \mapsto h3$, there must be two rules $r_{22} \in h2$ and $r_{31} \in h3$ and $(r_{22}, r_{31}) \in P$, where $r_{22} \mapsto r_{31}$. Moreover, according to the definition of a hypernode, we either have $r_{21} \mapsto r_{22}$ or $r_{21} = r_{22}$. Thus, if $h1 \mapsto h2 \mapsto ... \mapsto hk \mapsto h1$, then there must exist $r_{11}, r_{12} \in h1, r_{21}, r_{22} \in h2, ..., r_{k1}, r_{k2} \in hk$ where $r_{11} \mapsto r_{12} \mapsto r_{21} \mapsto r_{22} \mapsto ... \mapsto r_{k1} \mapsto r_{k2} \mapsto r_{11}$, meaning $r_{11}, r_{12}, r_{21}, r_{22}, ..., r_{k1}, r_{k2}$ belong to a single hypernode instead of different ones, which is contradictory to the original assumption. Similarly, for the strictly negative and positive/negative cases, an $HRDG$ is a DAG. □

C. Proof of correctness of Algorithm 1

The aim of this section is to show that Algorithm 1 correctly computes the semantics of a stratified Datalog program.

We begin byrecalling the definition of the semantics of a stratified Datalog program. Let P be a stratified Datalog program and let σ be one of its stratifications which induces a decomposition $P_1, \ldots, P_n$ of the program P. The semantics of P under the stratification σ with an extensional database EDB, denoted $P^\sigma(\mathrm{EDB})$, is defined as the limit of the increasing sequence $(I_k)_{k\in[0,n]}$ defined by:

$$\begin{array}{rcll} I_0 & = & \mathrm{EDB} & \\ I_i & = & I_{i-1}\cup P_i(I_{i-1} \mid \mathrm{edb}(P_i)) & \text{for } i\in[1,n] \end{array}$$

The semantics $P^{\mathrm{strat}}(\mathrm{EDB})$ of a stratified Datalog program P over an extensional database EDB is defined as $P^\sigma(\mathrm{EDB})$ for any stratification σ of P (see [1][Lemma 15.2.8]).

Lemma A.3. *For a stratified Datalog program P and an extensional database* EDB *over* $\mathrm{edb}(P)$*, Algorithm 1 computes* $P^{\mathrm{strat}}(\mathrm{EDB})$.

Proof. Let P be a stratified Datalog program and let HRDG be its associated hypergraph. Let $\mathrm{hn}_1,\dots,\mathrm{hn}_n$ be an enumeration of the hypernodes of HRDG following the topological ordering computed at the beginning of Algorithm 1 Algorithm 1 computes the sequence $(\mathrm{IDB}_{\mathrm{hn}_k})_{k\in[1,n]}$ defined by:

$$\mathrm{IDB}_{\mathrm{hn}_k} = \mathrm{hn}_k(\mathrm{EDB}_{\mathrm{hn}_k} \mid \mathrm{edb}(\mathrm{hn}_k))$$

where $\mathrm{EDB}_{\mathrm{hn}_k} = \bigcup_{(\mathrm{hn}',\mathrm{hn}_k)\in\mathrm{HRDG}} \mathrm{IDB}_{\mathrm{hn}'}\cup\mathrm{EDB}$. The value computed by Algorithm 1 is the union $\bigcup_{k\in[1,n]}\mathrm{IDB}_{\mathrm{hn}_k}$.

Let us first remark that the topological ordering $\mathrm{hn}_1,\dots,\mathrm{hn}_n$ of the hypernodes of HRDG naturally induces a stratification σ of P. For a rule r of P, the stratification $\sigma(r)$ is defined to be the unique $k\in[1,n]$ such that r belongs to hn_k. Assume towards a contradiction that σ is not a stratification of P. There are two possibilities:

- Either there would exists a positive edge (r,r') with $r\in\mathrm{hn}_i$ and $r'\in\mathrm{hn}_j$ with $i>j$ which contradicts the definition of a topological ordering.
- Or there would exists a negative edge (r,r') with $r\in\mathrm{hn}_i$ and $r'\in\mathrm{hn}_j$ with $i\geq j$. As we started with a topological ordering, it must be the case that $i=j$. As the edge is negative, this would imply that hn_i contains a cycle with a negative edge which contradicts the fact that P can be stratisfied.

The semantics of P under the stratification σ is the limit of the increasing sequence $(I_k)_{k\in[0,n]}$ defined by:

$$\begin{array}{rcll} I_0 & = & \mathrm{EDB} & \\ I_i & = & I_{i-1}\cup P_i(I_{i-1} \mid \mathrm{edb}(P_i)) & \text{for } i\in[1,n] \end{array}$$

We will prove by induction on $k\in[1,n]$ that:

$$I_k = \bigcup_{i\in[1,k]} \mathrm{IDB}_{\mathrm{hn}_i}\cup\mathrm{EDB}$$

For $k=1$, the result is immediate. For the induction step, assume that $I_k=\bigcup_{i\in[1,k]}\mathrm{IDB}_{\mathrm{hn}_i}\cup\mathrm{EDB}$ for some $k\geq 1$. To show that $I_{k+1}=\bigcup_{i\in[1,k+1]}\mathrm{IDB}_{\mathrm{hn}_i}\cup\mathrm{EDB}$, it is enough to show that

$$\mathrm{hn}(I_k \mid \mathrm{edb}(\mathrm{hn}_k)) = \mathrm{hn}(\mathrm{EDB}_{\mathrm{hn}_k} \mid \mathrm{edb}(\mathrm{hn}_k)).$$

Clearly $\mathrm{EDB}_{\mathrm{hn}_k}\subseteq I_k$. Now consider a fact in I_k for a predicate in $\mathrm{edb}(\mathrm{hn}_k)$. This fact either belongs to EDB or, by definition of HRDG to some hn' with $(\mathrm{hn}',\mathrm{hn}_k)\in\mathrm{HRDG}$. By definition of $\mathrm{EDB}_{\mathrm{hn}_k}$, it follows that $I_k \mid \mathrm{edb}(\mathrm{hn}_k) = \mathrm{EDB}_{\mathrm{hn}_k} \mid \mathrm{edb}(\mathrm{hn}_k)$. This concludes the induction proof.

We have established that:

$$P^{\mathrm{strat}}(\mathrm{EDB}) = P^\sigma(\mathrm{EDB}) = \bigcup_{i\in[1,k]}\mathrm{IDB}_{\mathrm{hn}_i}\cup\mathrm{EDB}.$$

To conclude the proof, it suffices to remark that $\mathrm{EDB}\subseteq\bigcup_{k\in[1,n]}\mathrm{IDB}_{\mathrm{hn}_k}$. Indeed $\mathrm{EDB}=\bigcup_{k\in[1,n]}\mathrm{EDB}\mid\mathrm{edb}(\mathrm{hn}_k)$ and for all $k\in[1,n]$, $\mathrm{EDB}\mid\mathrm{edb}(\mathrm{hn}_k)\subseteq\mathrm{hn}_k(\mathrm{EDB}_{\mathrm{hn}_k})$ by Lemma A.1.2 as $\mathrm{EDB}\subseteq\mathrm{EDB}_{\mathrm{hn}_k}$. ☐

D. Proof of Lemma 2

Lemma 2. *Let P be a semi-positive Datalog program and an extensional database EDB over* $\mathrm{edb}(P)$*. For B a subset of* $P(\mathrm{EDB})$*,*

$$P(\mathrm{EDB}) = \mathrm{SemiNaiveAlgorithm}(P,\mathrm{EDB}\cup B)$$

Proof. To establish this equality, we prove that the limit of the increasing sequence $(T^k_P(\mathrm{EDB}\cup B))_{k\geq1}$ is equal to $P(\mathrm{EDB})$.

We will show by induction on $k\geq 0$ that:

$$T^k_P(\mathrm{EDB})\subseteq T^k_P(\mathrm{EDB}\cup B)\subseteq T^k_P(P(\mathrm{EDB})).$$

For $k=0$, we do have that $\mathrm{EDB}\subseteq\mathrm{EDB}\cup B\subseteq P(\mathrm{EDB})$. For the induction step, assume that $T^k_P(\mathrm{EDB})\subseteq T^k_P(\mathrm{EDB}\cup B)\subseteq T^k_P(P(\mathrm{EDB}))$. By Lemma A.1.1, we have that:

$$\begin{array}{rcl} T^k_P(\mathrm{EDB})\mid\mathrm{edb}(P) & = & \mathrm{EDB}\mid\mathrm{edb}(P) \\ T^k_P(P(\mathrm{EDB}))\mid\mathrm{edb}(P) & = & P(\mathrm{EDB})\mid\mathrm{edb}(P) \\ & = & \mathrm{EDB}\mid\mathrm{edb}(P) \\ T^k_P(\mathrm{EDB}\cup B)\mid\mathrm{edb}(P) & = & \mathrm{EDB}\cup B\mid\mathrm{edb}(P) \end{array}$$

As $\mathrm{EDB}\subseteq\mathrm{EDB}\cup B\subseteq P(\mathrm{EDB})$, it follows that $\mathrm{EDB}\cup B\mid\mathrm{edb}(P)=\mathrm{EDB}\mid\mathrm{edb}(P)$. Hence, we have:

$$\begin{array}{rl} & T^k_P(\mathrm{EDB})\mid\mathrm{edb}(P) \\ = & T^k_P(\mathrm{EDB}\cup B)\mid\mathrm{edb}(P) \\ = & T^k_P(P(\mathrm{EDB}))\mid\mathrm{edb}(P) \\ = & \mathrm{EDB}\mid\mathrm{edb}(P) \end{array}$$

Hence we can apply Lemma A.1.3 to conclude the induction.

As $(T^k_P(\mathrm{EDB}))_{k\geq0}$ and $T^k_P(P(\mathrm{EDB}))$ share the same limit $P(\mathrm{EDB})$, we have that the limit of $(T^k_P(\mathrm{EDB}\cup B))_{k\geq0}$ is equal to $P(\mathrm{EDB})$. ☐

E. Proof of correctness in the case of rule insertions

We will now show that the semantics computed when adding rules is correct.

We start by recalling the procedure in the case of rule insertion. We start with a semi-positive stratified Datalog program P with its HRDG G. Furthermore, the semantics $\mathrm{IDB}_{\mathrm{hn}}$ for P of each hypernode hn of G is assumed to have been computed.

We consider a program P^t obtained by adding a new set of rules NR to P. We compute its HRDG G^t and the semantics IDB^t_{hn} of each of its hypernode hn as follows:

- using Algorithm 2, we compute G^t together with a set DIH of directly impacted hypernodes of G^t. For each hypernode $\mathrm{hn}\in\mathrm{DIH}$, Algorithm 2 provides a

bootstrapping value noted here $\mathrm{Boot}_{\mathrm{hn}}$ that will be used to speed up the computation of $\mathrm{IDB}^t_{\mathrm{hn}}$.
- using Algorithm 4, we compute a topological ordering on the set IH of hypernodes of G^t reachable for DIH.
- the semantics IDB^t_{hn} for hypernodes hn in G^t are computed as follows:
  - First, for hypernodes hn that do not belong to $\mathrm{DIH} \cup \mathrm{IH}$, $\mathrm{IDB}^t_{\mathrm{hn}}$ is taken to be equal to $\mathrm{IDB}_{\mathrm{hn}}$.
  - The values of $\mathrm{IDB}^t_{\mathrm{hn}}$ for $\mathrm{hn} \in \mathrm{DIH} \cup \mathrm{IH}$ are computed using the semi-naive algorithm using the topological ordering previously constructed as in Algorithm 1. For hypernodes hn in DIH the computation of the semi-naive algorithm is initialized using $\mathrm{Boot}_{\mathrm{hn}}$ instead of a empty set in the other case.

For an HRDG G and one of its hypernodes hn, we denote by $G{\downarrow_{\mathrm{hn}}}$ the HRDG obtained by restricting G to the hypernodes that can reach hn in G. The correctness of our procedure relies on the following key property:

Lemma A.4. *For a hypernode* hn *that does not belong to* $\mathrm{DIH} \cup \mathrm{IH}$,

$$G^t{\downarrow_{hn}} = G{\downarrow_{\mathrm{hn}}} \ .$$

Proof. As P' is obtained by adding rules from NR to P, the hypernodes of G^t are only of two form:
- a hypernode of G
- the union of a non-empty subset of NR with a possibly empty union of hypernodes of G.

Furthermore, the vertices in DIH are precisely the hypernodes of G^t containing a rule in NR.

Now consider a hypernode hn that does not belong to $\mathrm{IH} \cup \mathrm{DIH}$. As hn does not belong to DIH, hn does not contain a rule in NR and from what precedes hn belongs to G. As hn does not belong to IH, it is not reachable from any hypernode containing a rule in NR. Hence all hypernodes in $G^t{\downarrow_{\mathrm{hn}}}$ belong to G. It follows that $G{\downarrow_{\mathrm{hn}}} \subseteq G^t{\downarrow_{\mathrm{hn}}}$. Assume towards a contradiction that the inclusion is strict. That is to say that there exists a hypernode hn' that can reach hn in G but that does not belongs to G^t. This would imply that there exists a hypernode hn'' in G^t that contain hn' and at least one rule in NR. Hence hn'' would belong to DIH and as hn'' contains hn', it can reach hn also in G^t which contradicts the fact that all hypernodes that can reach hn in G^t also belong to G. □

Armed with Lemma A.4, we can show that our procedure is correct.

Lemma A.5. *The incremental update procedure is correct when adding rules.*

Proof. Clearly Algorithm 2 computes the HRDG G^t. Also Algorithm 4 is an adaptation to hypergraph of a the standard DFS based computation of a topological ordering. So we only need to show that the computation of the various $\mathrm{IDB}^t_{\mathrm{hn}}$ for $\mathrm{hn} \in G^t$ is correct.

We consider hypernodes in a topological ordering that extends the ordering computed by Algorithm 4 and show by induction that the computed value for $\mathrm{IDB}^t_{\mathrm{hn}}$ is correct.

If hn **does not belong to** $\mathrm{DIH} \cup \mathrm{IH}$**.** In this case, we take $\mathrm{IDB}^t_{\mathrm{hn}} = \mathrm{IDB}_{\mathrm{hn}}$. By Lemma A.4, we have that $G^t{\downarrow_{\mathrm{hn}}} = G{\downarrow_{\mathrm{hn}}}$. As $\mathrm{IDB}^t_{\mathrm{hn}}$ only depends on $G^t{\downarrow_{\mathrm{hn}}}$, it follows that $\mathrm{IDB}^t_{\mathrm{hn}}$ is indeed equal to $\mathrm{IDB}_{\mathrm{hn}}$.

If hn **belongs to** DIH**.** We take $\mathrm{EDB}^t_{\mathrm{hn}}$ to be the result of the semi-naive algorithm applied to hn with:

$$\mathrm{EDB}^t_{\mathrm{hn}} \cup \mathrm{Boot}_{\mathrm{hn}}$$

as input instance where:
- $\mathrm{EDB}^t_{\mathrm{hn}}$ is the union of all $\mathrm{EDB}^t_{\mathrm{hn}'}$ for $(\mathrm{hn}', \mathrm{hn}) \in G^t$,
- $\mathrm{Boot}_{\mathrm{hn}}$ is computed in Algorithm 2.

As by induction hypothesis, all the $\mathrm{EDB}^t_{\mathrm{hn}'}$ for $(\mathrm{hn}', \mathrm{hn}) \in G^t$ are correct, we only need to show that

$$\mathsf{SemiNaive}(hn, \mathrm{EDB}^t_{\mathrm{hn}}) = \mathsf{SemiNaive}(hn, \mathrm{EDB}^t_{\mathrm{hn}} \cup \mathrm{Boot}_{\mathrm{hn}}).$$

By Lemma 2 , it is enough to show that $\mathrm{Boot}_{\mathrm{hn}}$ is included in $\mathrm{hn}(\mathrm{EDB}^t_{\mathrm{hn}})$. By construction, $\mathrm{Boot}_{\mathrm{hn}}$ is a finite union of $\mathrm{IDB}_{\mathrm{hn}_1}, \ldots, \mathrm{IDB}_{\mathrm{hn}_k}$ with for $i \in [1, k]$, $\mathrm{hn}_i \subseteq \mathrm{hn}$. We will show that for all $i \in [1, k]$, $\mathrm{IDB}_{\mathrm{hn}_i} \subseteq \mathrm{IDB}^t_{\mathrm{hn}}$. Let $i \in [1, k]$. We have that $\mathrm{IDB}_{\mathrm{hn}_i} = \mathrm{hn}_i(\mathrm{EDB}_{\mathrm{hn}_i})$ and $\mathrm{IDB}^t_{\mathrm{hn}} = \mathrm{hn}(\mathrm{EDB}^t_{\mathrm{hn}})$. Using Lemma A.1.3, it is enough to shown that $\mathrm{EDB}_{\mathrm{hn}_i} \mid \mathrm{edb}(\mathrm{hn}_i) = \mathrm{EDB}^t_{\mathrm{hn}} \mid \mathrm{edb}(\mathrm{hn}_i)$. To see this, consider a relation $r \in \mathrm{edb}(\mathrm{hn}_i) \subseteq \mathrm{edb}(\mathrm{hn})$. Let hn be the hypernode in G^t containing r. We have $(\mathrm{hn}', \mathrm{hn}) \in G^t$ and hence hn' does not belong $\mathrm{DIH} \cup \mathrm{IH}$. We have already shown that $G^t{\downarrow_{\mathrm{hn}'}} = G{\downarrow_{\mathrm{hn}'}}$ and that $\mathrm{IDB}_{\mathrm{hn}'} = \mathrm{IDB}^t_{\mathrm{hn}'}$. The facts corresponding to the relation r in $\mathrm{EDB}_{\mathrm{hn}_i}$ are contained in $\mathrm{IDB}_{\mathrm{hn}'}$. Similarly the facts corresponding to the relation r in $\mathrm{EDB}^t_{\mathrm{hn}}$ are contained in $\mathrm{IDB}^t_{\mathrm{hn}'} = \mathrm{IDB}_{\mathrm{hn}'}$.

If hn **belongs to** IH. $\mathrm{EDB}^t_{\mathrm{hn}}$ is computed exactly as in Algorithm 1 which has been shown to produce the correct semantics in Lemma A.3. □

F. Proof of correctness in the case of rule deletions

We will now show that the semantics computed when rules are removed is correct.

We start by recalling the procedure in the case of rule deletion. We start with a semi-positive stratified Datalog program P with its HRDG G. Furthermore, the semantics $\mathrm{IDB}_{\mathrm{hn}}$ for P of each hypernode hn of G is assumed to have been computed.

We consider a program P^t obtained by removing a set of rules DR from P. We compute its HRDG G^t and the semantics IDB^t_{hn} of each of its hypernode hn as follows:
- using Algorithm 3 , we compute G^t together with a set DIH of directly impacted hypernodes of G^t.
- using Algorithm 4 , we compute a topological ordering on the set IH of hypernodes of G^t reachable for DIH.
- the semantics IDB^t_{hn} for hypernodes hn in G^t are computed as follows:
  - First, for hypernodes hn that do not belong to $\mathrm{DIH} \cup \mathrm{IH}$, $\mathrm{IDB}^t_{\mathrm{hn}}$ is taken to be equal to $\mathrm{IDB}_{\mathrm{hn}}$.

– The values of $\mathrm{IDB}^t_{\mathrm{hn}}$ for $\mathrm{hn} \in \mathrm{DIH} \cup \mathrm{IH}$ are computed using the semi-naive algorithm using the topological ordering previously constructed as in Algorithm 1.

The key property of Algorithm 3 is given by the following lemma.

Lemma A.6. *For a hypernode* hn *that does not belong to* $\mathrm{DIH} \cup \mathrm{IH}$,

$$G^t{\downarrow}_{hn} = G{\downarrow}_{\mathrm{hn}} .$$

Proof. As P' is obtained by removing rules from P, the hypernodes of G^t are only of two form:

- a hypernode of G,
- or strict subset of hypernodes of G that contain rules in DR.

Furthermore, the vertices in DIH are precisely the hypernodes of G^t that are strict subsets of hypernodes of G and the hypernodes of G that are successor in G of hypernodes containing a rule in DR.

Now consider an hypernode hn that does not belong to $\mathrm{IH} \cup \mathrm{DIH}$. As hn does not belong to DIH, hn belongs to G. It suffices to show that $G{\downarrow}_{\mathrm{hn}}$ only contains hypernodes without rules in DR. Toward a contradiction, assume it is not the case. Let hn' be a predecessor of hn in G containing a rule in DR with the smallest distance to hn. By minimality of the distance, there exists a successor hn'' of hn' which is a predecessor of hn and which belong to G. By construction Algorithm 2, hn'' belongs to DIH and hence hn belongs to $\mathrm{DIH} \cup \mathrm{IH}$ which brings the contradiction. As P' is obtained by removing rules from P, the desired equality follows from the fact that $G{\downarrow}_{\mathrm{hn}}$ only contains hypernodes without rules in DR. □

Lemma A.7. *The incremental update procedure is correct when deleting rules.*

Proof. Clearly Algorithm 3 computes the HRDG G^t. Also Algorithm 4 is an adaptation to hypergraph of a the standard DFS based computation of a topological ordering. So we only need to show that the computation of the various $\mathrm{IDB}^t_{\mathrm{hn}}$ for $\mathrm{hn} \in G^t$ is correct.

We consider hypernodes in a topological ordering that extends the ordering computed by Algorithm 4 and show by induction that the computed value for $\mathrm{IDB}^t_{\mathrm{hn}}$ is correct.

If hn **does not belong to** $\mathrm{DIH} \cup \mathrm{IH}$**.** In this case, we take $\mathrm{IDB}^t_{\mathrm{hn}} = \mathrm{IDB}_{\mathrm{hn}}$. By Lemma A.6, we have that $G^t{\downarrow}_{\mathrm{hn}} = G{\downarrow}_{\mathrm{hn}}$. As $\mathrm{IDB}^t_{\mathrm{hn}}$ only depends on $G^t{\downarrow}_{\mathrm{hn}}$, it follows that $\mathrm{IDB}^t_{\mathrm{hn}}$ is indeed equal to $\mathrm{IDB}_{\mathrm{hn}}$.

If hn **belongs to** $\mathrm{DIH} \cup \mathrm{IH}$**.** $\mathrm{IDB}^t_{\mathrm{hn}}$ is computed exactly as in Algorithm 1 which has been shown to produce the correct semantics in Lemma A.3.

□

G. Proof of Lemma 3

Lemma 3. *The incremental approach designed for rule amendments is correct.*

Proof. This is a direct consequence of Lemma A.5 and Lemma A.7. □