Technical Appendix

LUIGI BELLOMARINI, Banca d'Italia
DAVIDE BENEDETTO, Università Roma Tre
MATTEO BRANDETTI, TU Wien
EMANUEL SALLINGER, TU Wien & University of Oxford

1 FURTHER WORKING DEFINITIONS

Given a fact $\mathbf{v}$ in the chase graph $\mathcal{G}(D, \Sigma)$, the *subgraph*$(\mathcal{G}, \mathbf{v})$ is the subgraph of $\mathcal{G}(D, \Sigma)$ induced by all the descendants of $\mathbf{v}$. Given a fact $\mathbf{v}$ in the warded forest $\mathcal{W}(\mathcal{G})$, the *subtree*$(\mathcal{W}, \mathbf{v})$ is the subtree induced by all the descendants of $\mathbf{v}$ in $\mathcal{W}$. Observe that the track of a fact in the warded forest $\mathcal{W}$ can be defined also as follows. Given two facts x and y of $\mathcal{G}(\Sigma, D)$, we define $\mathbf{b} = \mathbf{track}(\mathbf{a})$ as the $d(\mathbf{b}, \mathbf{a})$-maximal fact $\mathbf{b}$ of $\mathcal{G}$ s.t. $\mathbf{a} \in$ *subtree*$(\mathcal{W}, \mathbf{b})$, where $d(x, y)$ is the graph distance of x and y.

2 DEFINITIONS OF HARMLESS EGDS

We provide two definitions of harmless EGDs and prove their equivalence.

Definition 3.1. *Given a set of TGDs and EGDs $\Sigma = \Sigma_T \cup \Sigma_E$ over a schema $\mathbf{S}$, we define Σ_E as* harmless *if for every database D over $\mathbf{S}$, if chase(D, Σ) does not fail, there exists a homomorphism h mapping chase(D, Σ_T) onto chase(D, Σ).*

As stated by this definition, harmless EGDs individuate the cases where a universal model for D and Σ can be obtained as a special case (i.e., an image) of a universal model for D and Σ_T. This notion can be equivalently formulated in terms of query answering.

Definition 3.2. *Given a set of TGDs and EGDs $\Sigma = \Sigma_T \cup \Sigma_E$ over a schema $\mathbf{S}$, we define Σ_E as harmless if for every database D over $\mathbf{S}$, if chase(D, Σ) does not fail, for every BCQ Q over $\mathbf{S}$ there exists a partial homomorphism h mapping chase(D, Σ_T) to chase(D, Σ), such that chase$(D, \Sigma) \models Q$ iff $h($chase$(D, \Sigma_T)) \models Q$.*

The reason why we give two definitions is that we often use the simpler Definition 3.1, while the style of Definition 3.2 is often used in related work.

Lemma 1. *Definitions 3.1 and 3.2 are equivalent.*

Proof. *We first show that Definition 3.1 implies Definition 3.2, and then that Definition 3.2 implies Definition 3.1.*

(Def 3.1 $\Rightarrow$ Def 3.2) By Def 3.1, as Σ_E is harmless, there exists a homomorphism h mapping *chase*(D, Σ_T) onto *chase*(D, Σ). To prove Def 3.2, let us separately show that for every BCQ Q, there exists a partial homomorphism h' mapping *chase*(T, Σ_T) to *chase*(T, Σ) such that *chase*$(D, \Sigma) \models Q \Rightarrow h'($*chase*$(D, \Sigma_T)) \models Q$ and *chase*$(D, \Sigma) \models Q \Leftarrow h'($*chase*$(D, \Sigma_T)) \models Q$.
($\Rightarrow$) Let us choose $h' = h$. By Def 3.1, *chase*(D, Σ) coincides with $h'($*chase*$(D, \Sigma_T))$. Then, since by Def 3.2 we have that *chase*$(D, \Sigma) \models Q$, it follows that $h'($*chase*$(D, \Sigma_T)) \models Q$.
($\Leftarrow$) Let us choose $h' = h$. By Def 3.1, *chase*(D, Σ) coincides with $h'($*chase*$(D, \Sigma_T))$. Then, since by Def 3.2 we have that $h'($*chase*$(D, \Sigma_T)) \models Q$, it follows that *chase*$(D, \Sigma) \models Q$.

(Def 3.2 ⇒ Def 3.1) Let us construct a BCQ Q as the conjunction of a set of atoms built in the following way: consider each fact $\boldsymbol{f}$ of $chase(D, \Sigma)$ and let $k : \boldsymbol{N} \rightarrow \boldsymbol{V}$ be a function that maps a set of labelled nulls into a set of variables. For each $\boldsymbol{f}$ construct an atom $\boldsymbol{a}$ preserving its constants and replacing its labelled nulls with fresh variables or, if already mapped, with existing images of k. If a new mapping is applied, then add it to k. Finally, add $\boldsymbol{a}$ to Q. By Def 3.2, there exists a partial homomorphism h mapping $chase(D, \Sigma_T)$ to $chase(D, \Sigma)$, such that $chase(D, \Sigma) \models Q$ iff $h(chase(D, \Sigma_T)) \models Q$. To prove Def 3.1, we need to show that, for the homomorphism h for such a query Q, we have that $chase(D, \Sigma)$ and $h(chase(D, \Sigma_T))$ coincide. To this end, we need to show that h is surjective (it maps $chase(D, \Sigma_T)$ onto $chase(D, \Sigma)$) and total (it maps all the elements of $chase(D, \Sigma_T)$). Let us proceed separately for the two properties.

Surjectivity. By Def 3.2(⇒) and given that $chase(D, \Sigma) \models Q$ by construction, it follows that $h(chase(D, \Sigma_T)) \models Q$. Then there exists at least a homomorphism θ such that for every atom $\boldsymbol{\varphi}(\mathbf{x}) \in Q$, it holds $\theta(\boldsymbol{\varphi}(\mathbf{x})) \in h(chase(D, \Sigma_T))$. Therefore h is surjective.

Totality. Let us proceed by contradiction and assume h is partial. Let Q' be the query that contains all the atoms that have a mapping in $chase(D, \Sigma_T)$ but are not mapped by h. By Def 3.2, there exists a homomorphism h' such that $h'(chase(D, \Sigma_T)) \models Q'$ iff $chase(D, \Sigma) \models Q'$. Now, as $chase(D, \Sigma_T) \models Q'$ by construction, by Def 3.2(⇐), it follows that $chase(D, \Sigma) \models Q'$. Since we have proved h is surjective, then it covers the facts of Q' in $chase(D, \Sigma)$. The only way for h to cover the atoms of Q' in $chase(D, \Sigma)$ is to map them from the ones of Q' in $chase(D, \Sigma_T)$, as h is functional. This leads a contradiction, since the atoms of Q' have been excluded from the domain of h. Therefore h cannot be partial and is then total. □

3 PROOF OF THEOREM 3.3

Theorem 3.3. *Given a set of TGDs and EGDs $\Sigma = \Sigma_T \cup \Sigma_E$, the problem of deciding whether Σ_E is harmless is undecidable.*

Proof. We proceed by reduction from Boolean conjunctive query answering under arbitrary TGDs, known to be undecidable even for simple atomic queries [6], to the co-problem of harmlessness. Let $Q : q \leftarrow r(c_1, c_2)$ be an arbitrary atomic query, D a database and Σ a set of rules such that $r(c_1, c_2) \in chase(D, \Sigma)$ and $r(c_1, c_2) \notin D$. We construct a new set of rules Σ' starting from Σ and adding to it: a rule $\top \rightarrow f$ for each fact $f \in D$ (where $\top$ is the constant *true*) plus the following TGDs (over predicates r^* and s^*, not appearing in Σ): $\sigma_1 : r(c_1, c_2), r^*(x, x) \rightarrow s^*(x, x)$ and $\sigma_2 : r(c_1, c_2), s^*(x, y) \rightarrow \exists z\, r^*(x, z)$ and the EGD $\eta : s^*(x, y), r^*(x, z) \rightarrow x = z$. Now, we argue that $chase(D, \Sigma) \models Q$ iff Σ'_E is not harmless.

(⇒) Assume that $D \cup \Sigma \models Q$. We have to show that Σ'_E is not harmless. Consider the database $D' = \{s^*(1, 2)\}$. It is clear that $D' \models \eta$ and $chase(D, \Sigma')$ does not fail by construction. Given that $D \cup \Sigma \models Q$ holds, by assumption, we also know that $r(c_1, c_2) \in chase(D, \Sigma)$. Hence $r(c_1, c_2) \in chase(D', \Sigma)$. Thus σ_2 can fire in $chase(D', \Sigma')$, it activates η that, in turn, activates σ_1 and so finally $s^*(1, 1) \in chase(D', \Sigma')$. Now let us consider $chase(D', \Sigma' \setminus \{\eta\})$. We have that $s^*(1, 1) \notin chase(D', \Sigma' \setminus \{\eta\})$ and no facts for s^* are originally appearing in Σ. Hence there is no homomorphism h from $chase(D', \Sigma' \setminus \{\eta\})$ to $chase(D', \Sigma')$, and Σ'_E is therefore not harmless.

(⇐) If Σ'_E is not harmless, it is easy to verify that $r(c_1, c_2) \in chase(D, \Sigma)$: in fact, if it were not the case, σ_1 and σ_2 could not be activated, contradicting our hypothesis. □

4 EXAMPLES OF NON-HARMLESS EGDS

To complement our technical results we present two examples where the EGDs, instead, exhibit a harmful behaviour.

Example 4.1.

$$element(x) \rightarrow \exists z\ comp(x, z) \tag{σ_1}$$

$$rest(x, y) \rightarrow \exists z\ comp(x, z), comp(y, z) \tag{σ_2}$$

$$comp(x, z'), comp(x, z'') \rightarrow z' = z'' \tag{η_1}$$

$$comp(x, z), comp(y, z) \rightarrow siblings(x, y) \tag{σ_3}$$

Here, elements x and y are clustered together in a set z if a binary relation rest holds for the pair $\langle x, y\rangle$ (σ_2). Each element belongs to one set only (η_1). We then mark as siblings all the pairs of elements in the same set z (σ_3). ■

Given a database $D = \{element(a), element(b), element(c), rest(a, b), rest(b, c)\}$, if we do not consider η_1, the only triggers for σ_3 would be the facts produced by σ_2, i.e., the pairs $\langle x, y\rangle$ for which *rest* is in D. For each such pair, a new set z would be created by σ_2, thus triggering the creation of *sibling* facts by σ_3, i.e., $siblings(a, b)$ and $siblings(b, c)$. What we wish to model here with η_1 is a form of transitivity of the *rest* relation: if $rest(a, b)$, $rest(b, c)$ holds, we would like a, b, c to be all grouped within the same set. The EGD encodes this behaviour by unifying the sets of elements to which an element x belongs. So actually, the EGD produces facts that act as triggers for σ_3, which yields the fact $siblings(a, c)$ that could not be produced otherwise. A homomorphism as defined in Definition 3.1 is clearly impossible to find.

Example 4.2.

$$a(x, y, w) \rightarrow \exists z\ b(x, y, z) \tag{σ_1}$$

$$a(x, y, w) \rightarrow \exists z\ b(x, z, w) \tag{σ_2}$$

$$b(x, y', w'), b(x, y'', w'') \rightarrow y' = y'', w' = w'' \tag{η_1}$$

$$b(x, y, z) \rightarrow f(x, y, z) \tag{σ_3}$$

$$b(x, x, x) \rightarrow f(x, x, x) \tag{σ_4}$$

A data fusion case where the source a is artificially split (σ_1 and σ_2) and then unified by η_1 and copied to f. ■

With reference to Example 4.2, consider $D = \{a(1, 1, 1)\}$ and the BCQ $Q : q \leftarrow f(1, 1, 1)$. We have that $chase(D, \Sigma) \models Q$, as it contains $f(1, 1, 1)$, generated by either σ_3 or σ_4. The chase $chase(D, \Sigma_T)$ contains $f(1, 1, z_1)$ and $f(1, z_2, 1)$, generated by σ_3 and thus, with a homomorphism $h = \{z_1 \rightarrow 1,\ z_2 \rightarrow 1\}$ from $chase(D, \Sigma_T)$ to $chase(D, \Sigma)$, we can satisfy Q. It easy to see that unless D violates η_1 (x is not a key), a homomorphism can be found for every D. Therefore Σ_E is harmless. Nevertheless, η_1 exhibits a potentially harmful behaviour, since it generates the only triggering facts for σ_4. Fortunately, the facts generated by σ_4 are by construction less general than (i.e., an image of) those generated by σ_3. In fact, if we removed σ_3, then $chase(D, \Sigma_T)$ would not contain any facts for f and therefore no homomorphism h would exist. In this case, Σ_E would be harmful. While we cannot check that potentially harmful EGDs like σ_4 are actually generating facts that could not be obtained otherwise, we would like a syntactic condition to frame out all these cases and to be sufficient to witness harmlessness. Example 4.2 inspires an interesting intuition concerning the relationship between labelled nulls and the variables unified in the EGDs. For harmlessness, we can tolerate that atom terms affected by an EGD appear in the body of a TGD (e.g., the second and the third term of b appearing in the body of

σ_3 and σ_4), yet those terms should not be determinant for the applicability of a TGD. It is the case of σ_3, whilst this does not happen in σ_4, where the values of affected terms are used to decide whether the TGD must fire.

5 TAINTEDNESS

Definition 3.4. *Given a set of rules Σ, a position $p[i]$ is inductively defined as tainted if: (i) $p[i]$ is affected and Σ contains an EGD $\eta = \boldsymbol{\phi}(\mathbf{x}) \rightarrow x_i = x_j$ such that $p[i]$ is the position of x_i (resp. x_j) in body(η) and x_i (resp. x_j) appears only in affected positions of body(η) (it is harmful); (ii) for some TGD $\sigma \in \Sigma$ and some variable $v \in body(\sigma) \cap head(\sigma)$, v appears in a tainted position in body(σ) (resp. head(σ)) and in position $p[i]$ in head(σ) (resp. body(σ)). A variable appearing in a tainted position of a rule σ is named tainted variable (wrt σ). Let tainted(σ) be the set of all the tainted variables in the body of σ.*

6 PROOF OF THEOREM 3.5

Theorem 3.5. *Let $\Sigma = \Sigma_T \cup \Sigma_E$ be a set of TGDs Σ_T and EGDs Σ_E over schema $\mathbf{S}$. The set Σ_E is harmless with respect to Σ if (safe taintedness) for every dependency $\sigma \in \Sigma$: (i) every variable $v \in tainted(\sigma)$ appears only once in body(σ), and, (ii) there are no constants appearing in tainted positions (i.e., there are no ground tainted positions).*

Proof. Assume that (safe taintedness) for every dependency $\sigma \in \Sigma$: (i) every variable $v \in \mathit{tainted}(\sigma)$ appears only once in $\mathit{body}(\sigma)$, and, (ii) there are no constants appearing in tainted positions (i.e., there are no ground tainted positions). We need to show that Σ is harmless. By Definition 3.1, we thus need to prove that for every database D of schema $\mathbf{S}$, if $\mathit{chase}(D, \Sigma)$ does not fail, there exists a homomorphism mapping $\mathit{chase}(D, \Sigma_T)$ onto $\mathit{chase}(D, \Sigma)$.

No tainted positions. Let us first analyse the trivial cases. If $\Sigma_E = \emptyset$, the chase cannot fail by construction and it is sufficient to choose the empty homomorphism $h = \emptyset$ to fulfil Definition 3.1. Therefore Σ_E is harmless. If $\Sigma_E \neq \emptyset$ but there are no tainted positions in Σ, by Definition of taintedness, either there are no affected positions in Σ, or there are no harmful variables in the body of any EGD $\eta : \boldsymbol{\phi}(\mathbf{x}) \rightarrow x_i = x_j$ of Σ_E. In either case, in the chase construction, EGDs only equate ground values. Therefore $\mathit{chase}(D, \Sigma)$ can fail only for a hard violation of Σ_E and since no EGD in Σ_E binds any labelled null to a ground value, it is sufficient to choose the empty homomorphism $h = \emptyset$ to fulfil Definition 3.1.

Tainted positions. Let us now account for the presence of tainted positions in TGDs $\sigma : \boldsymbol{\phi}(\mathbf{x}) \rightarrow \exists \mathbf{z}\, \boldsymbol{\psi}(\mathbf{y}, \mathbf{z})$ of Σ_T. We start to prove that Definition 3.1 holds, we must show that if $\mathit{chase}(D, \Sigma)$ does not fail, there exists a homomorphism h mapping $\mathit{chase}(D, \Sigma_T)$ onto $\mathit{chase}(D, \Sigma)$, and so that $\mathit{chase}(D, \Sigma) = h(\mathit{chase}(D, \Sigma_T))$. We proceed by proving the two directions separately, i.e., cases (a) $\mathit{chase}(D, \Sigma) \subseteq h(\mathit{chase}(D, \Sigma_T))$, and (b) $h(\mathit{chase}(D, \Sigma_T)) \subseteq \mathit{chase}(D, \Sigma)$.

Case (a): $\mathit{chase}(D, \Sigma) \subseteq h(\mathit{chase}(D, \Sigma_T))$ holds. To prove (a), we proceed by induction on $\mathit{chase}(D, \Sigma)$ and show that it is possible to incrementally build h such that it holds as an induction invariant that for every conjunction of facts $\boldsymbol{\Psi}(\mathbf{y}, \mathbf{z}) \in \mathit{chase}(\Sigma, D)$ there exists a conjunction of facts $\boldsymbol{\Psi}^{\boldsymbol{T}}(\mathbf{y}, \mathbf{z}) \in \mathit{chase}(\Sigma_T, D)$ which is mapped onto $\boldsymbol{\Psi}(\mathbf{y}, \mathbf{z})$ by h.

(a.1) *Base case.* Let $\boldsymbol{\Psi_1}(\mathbf{y}, \mathbf{z})$ be a conjunction of facts produced by one single initial chase step $\boldsymbol{\varphi_0}(\mathbf{x}) \xrightarrow{\sigma_\theta} \boldsymbol{\Psi_1}(\mathbf{y}, \mathbf{z})$ of $\mathit{chase}(D, \Sigma)$, where σ_θ denotes that a TGD σ has been applied by the first TGD chase step, with a triggering homomorphism θ from $\mathbf{x}$ to D. Since $\boldsymbol{\varphi_0}(\mathbf{x})$ is in D, the chase step $\boldsymbol{\varphi_0}(\mathbf{x}) \xrightarrow{\sigma_\theta} \boldsymbol{\Psi}_{\mathbf{1}}^{\boldsymbol{T}}(\mathbf{y}, \mathbf{z})$ of $\mathit{chase}(D, \Sigma_T)$ is activated and hence it holds that $\boldsymbol{\Psi_1}(\mathbf{y}, \mathbf{z}) = \boldsymbol{\Psi}_{\mathbf{1}}^{\boldsymbol{T}}(\mathbf{y}, \mathbf{z})$ by construction. We initialize $h_0 = \emptyset$.

(a.2) *Inductive case.* We now consider a conjunction of facts $\boldsymbol{\Psi_n}(\mathbf{y}, \mathbf{z}) \in \mathit{chase}^n(D, \Sigma)$ generated by the first n chase steps, and want to prove the existence of a conjunction of facts $\boldsymbol{\Psi}_{\boldsymbol{n}}^{\boldsymbol{T}}(\mathbf{y}, \mathbf{z}) \in \mathit{chase}(D, \Sigma_T)$ such that $h_n(\boldsymbol{\Psi}_{\boldsymbol{n}}^{\boldsymbol{T}}(\mathbf{y}, \mathbf{z})) = \boldsymbol{\Psi_n}(\mathbf{y}, \mathbf{z})$. Two cases are possible.

- *EGD chase step.* $\boldsymbol{\Psi_n}(\mathbf{y},\mathbf{z})$ has been generated as a consequence of the bindings induced in $chase(D,\Sigma)$ by an *EGD chase step*: $\boldsymbol{\varphi_{n-1}}(\mathbf{x}) \xrightarrow{\sigma_\theta} x_i = x_j$. By inductive hypothesis, we assume that for $\boldsymbol{\Psi_{n-1}}(\mathbf{y},\mathbf{z}) \in chase(D,\Sigma)$ there is a conjunction of facts $\boldsymbol{\Psi^T_{n-1}}(\mathbf{y},\mathbf{z}) \in chase(D,\Sigma_T)$ such that $\boldsymbol{\Psi_{n-1}}(\mathbf{y},\mathbf{z}) = h_{n-1}(\boldsymbol{\Psi^T_{n-1}}(\mathbf{y},\mathbf{z}))$. We now let $h_n = h_{n-1} \circ \{x_i \rightarrow x_j\}$, that is, we update h, by composing it with the assignments introduced by the EGD chase step. By construction, it follows $h_n(\boldsymbol{\psi^T_n}(\mathbf{y},\mathbf{z})) = \boldsymbol{\psi_n}(\mathbf{y},\mathbf{z})$.
- *TGD chase step.* $\boldsymbol{\Psi_n}(\mathbf{y},\mathbf{z})$ has been generated by a *TGD chase step* $\boldsymbol{\varphi_{n-1}}(\mathbf{x}) \xrightarrow{\sigma_\theta} \boldsymbol{\Psi_n}(\mathbf{y},\mathbf{z})$. By inductive hypothesis, we assume that for $\boldsymbol{\varphi_{n-1}}(\mathbf{x}) \in chase(D,\Sigma)$ there is a conjunction of facts $\boldsymbol{\varphi^T_{n-1}}(\mathbf{x}) \in chase(D,\Sigma_T)$ such that $\boldsymbol{\varphi_{n-1}}(\mathbf{x}) = h_{n-1}(\boldsymbol{\varphi^T_{n-1}}(\mathbf{x}))$. We show that if $\theta(\boldsymbol{\phi}(\mathbf{x})) \subseteq chase^{n-1}(D,\Sigma)$, it follows $\theta(\boldsymbol{\phi}(\mathbf{x})) \subseteq chase^{n-1}(D,\Sigma_T)$. To prove this, we argue that if θ maps $\boldsymbol{\phi}(\mathbf{x})$ to $\boldsymbol{\varphi_{n-1}}(\mathbf{x})$, then it also maps $\boldsymbol{\phi}(\mathbf{x})$ to $\boldsymbol{\varphi^T_{n-1}}(\mathbf{x})$. As, $\boldsymbol{\varphi_{n-1}}(\mathbf{x}) = h_{n-1}(\boldsymbol{\varphi^T_{n-1}}(\mathbf{x}))$, the only two possibilities that θ applies to $\boldsymbol{\varphi_{n-1}}(\mathbf{x})$ and not to $\boldsymbol{\varphi^T_{n-1}}(\mathbf{x})$ are that: (p1) h_{n-1} replaces a labelled null v_i in position $\boldsymbol{\phi}[i]$ of $\boldsymbol{\varphi^T_{n-1}}(\mathbf{x})$ with a constant and $\boldsymbol{\phi}[i]$ is ground in $\boldsymbol{\phi}(\mathbf{x})$; (p2) h_{n-1} replaces a labelled null v_i in position $\boldsymbol{\phi}[i]$ of $\boldsymbol{\varphi^T_{n-1}}(\mathbf{x})$ with another variable null v_j, already appearing in position $\boldsymbol{\phi}[j]$, with $i \neq j$, and $\boldsymbol{\phi}(\mathbf{x})$ contains the same variable in positions $\boldsymbol{\phi}[i]$ and $\boldsymbol{\phi}[j]$.

  Possibilities (p1) and (p2) do not occur in any of our cases, which we show next. In the base case, $h = \emptyset$. In the TGD inductive case, h is not extended with assignments to constants, nor is it extended with assignments to labelled nulls already appearing in any atom (as we specify at the end this point). Finally, let us focus on the EGD inductive case (see previous point). If $\boldsymbol{\phi}[i]$ is ground in $\boldsymbol{\phi}(\mathbf{x})$, by safe taintedness, $\boldsymbol{\phi}[i]$ cannot be tainted, therefore, no EGD chase step can map any labelled null appearing in $\boldsymbol{\phi}[i]$ of $\boldsymbol{\varphi^T_{n-1}}(\mathbf{x})$ into a constant, and thus h cannot contain such an assignment. If $\boldsymbol{\phi}[i]$ is tainted and is not ground in $\boldsymbol{\phi}(\mathbf{x})$, by safe taintedness, there cannot be any other positions $\boldsymbol{\phi}[j]$, with $i \neq j$ where $\boldsymbol{\phi}(\mathbf{x})$ has the same variable. Therefore neither (p1) nor (p2) ever occur.

  We now build h_n by extending h_{n-1} as $h_n = h_{n-1} \cup \{z^T_i \rightarrow z_i, \ldots\}$, where $z^T \rightarrow z_i$ denotes the correspondences between fresh labelled nulls of $\mathbf{z}$ in $\boldsymbol{\Psi^T_n}(\mathbf{y},\mathbf{z})$ resp. $\boldsymbol{\Psi}(\mathbf{y},\mathbf{z})$. It follows that $h_n(\boldsymbol{\Psi^T_n}(\mathbf{y},\mathbf{z})) = \boldsymbol{\Psi_n}(\mathbf{y},\mathbf{z})$.

Case (b): $h(chase(D,\Sigma_T)) \subseteq chase(D,\Sigma)$ holds. To prove (b), we proceed by induction on $chase(D,\Sigma_T)$ and show that it is possible to incrementally build a homomorphism h such that it holds as an induction invariant that for every conjunction of facts $\boldsymbol{\Psi}^T(\mathbf{y},\mathbf{z}) \in chase(\Sigma_T,D)$ there exists a conjunction of facts $\boldsymbol{\Psi}(\mathbf{y},\mathbf{z}) \in chase(\Sigma,D)$ onto which $\boldsymbol{\Psi}^T(\mathbf{y},\mathbf{z})$ is mapped by h.

(b.1) *Base case.* Let $\boldsymbol{\Psi^T_1}(\mathbf{y},\mathbf{z})$ be a conjunction of facts produced by one single initial chase step $\boldsymbol{\varphi^T_0}(\mathbf{x}) \xrightarrow{\sigma_\theta} \boldsymbol{\Psi^T_1}(\mathbf{y},\mathbf{z})$ of $chase(D,\Sigma_T)$. Since $\boldsymbol{\varphi^T_0}(\mathbf{x})$ is in D, the chase step $\boldsymbol{\varphi^T_0}(\mathbf{x}) \xrightarrow{\sigma_\theta} \boldsymbol{\Psi_1}(\mathbf{y},\mathbf{z})$ of $chase(D,\Sigma)$ is activated and hence it holds $\boldsymbol{\Psi^T_1}(\mathbf{y},\mathbf{z}) = \boldsymbol{\Psi_1}(\mathbf{y},\mathbf{z})$ by construction. We initialize $h_0 = \emptyset$.

(b.2) *Inductive case.* We now consider a conjunction of facts $\boldsymbol{\Psi^T_n}(\mathbf{y},\mathbf{z}) \in chase^n(D,\Sigma_T)$ generated by the first n chase steps, and want to prove the existence of a conjunction of facts $\boldsymbol{\Psi_n}(\mathbf{y},\mathbf{z}) \in chase(D,\Sigma)$ such that $\boldsymbol{\Psi_n}(\mathbf{y},\mathbf{z}) = h_n(\boldsymbol{\Psi^T_n}(\mathbf{y},\mathbf{z}))$. For $\boldsymbol{\Psi^T_n}(\mathbf{y},\mathbf{z})$ to exist, it must have been generated by a *TGD chase step* $\boldsymbol{\varphi^T_{n-1}}(\mathbf{x}) \xrightarrow{\sigma_\theta} \boldsymbol{\Psi^T_n}(\mathbf{y},\mathbf{z})$. By inductive hypothesis, we assume that for $\boldsymbol{\varphi^T_{n-1}}(\mathbf{x}) \in chase(D,\Sigma_T)$ there is a conjunction of facts $\boldsymbol{\varphi_{n-1}}(\mathbf{x}) \in chase(D,\Sigma)$ such that $h_{n-1}(\boldsymbol{\varphi^T_{n-1}}(\mathbf{x})) = \boldsymbol{\varphi_{n-1}}(\mathbf{x})$. If $\theta(\boldsymbol{\phi}(\mathbf{x})) \subseteq chase^{n-1}(D,\Sigma_T)$, there exists a triggering homomorphism $\theta' = h_{n-1} \circ \theta$ such that $\theta'(\boldsymbol{\phi}(\mathbf{x})) \subseteq chase^{n-1}(D,\Sigma)$. We now build h_n by extending h_{n-1} as $h_n = h_{n-1} \cup \{z^T_i \rightarrow z_i, \ldots\}$, where $z^T \rightarrow z_i$ denotes the correspondences between fresh labelled nulls of $\mathbf{z}$ in $\boldsymbol{\Psi^T_n}(\mathbf{y},\mathbf{z})$ resp. $\boldsymbol{\Psi}(\mathbf{y},\mathbf{z})$. It follows $\boldsymbol{\Psi_n}(\mathbf{y},\mathbf{z}) = h_n(\boldsymbol{\Psi^T_n}(\mathbf{y},\mathbf{z}))$.

Having shown both cases (a) and (b), this concludes our proof that Definition 3.1 holds. □

7 PROOF OF THEOREM 3.6

Theorem 3.6. *Given a set Σ_T of warded TGDs and a database D, for every BCQ Q, it holds that $chase^W(D, \Sigma_T) \models Q$ iff $chase(D, \Sigma_T) \models Q$, i.e., $chase^W(D, \Sigma_T)$ is BCQ-equivalent to $chase(D, \Sigma_T)$*

Proof. We prove this result for atomic queries and then we extend it to BCQs. We assume the absence of harmful joins in Σ_T (i.e., joins on positions possibly containing labelled nulls) as it is always possible to perform a normalization [3].
($\Rightarrow$) This proposition holds by construction since $chase^W(D, \Sigma_T) \subseteq chase(D, \Sigma_T)$.
($\Leftarrow$) Let us proceed by contradiction and assume there exists an atomic query Q such that $chase(D, \Sigma) \models Q$ and $chase^W(D, \Sigma) \not\models Q$. Thus, there exists a homomorphism θ such that $\theta(Q) \in chase(D, \Sigma)$ and there does not exist any homomorphism θ' such that $\theta'(Q) \in chase^W(D, \Sigma)$. Consider the tree consisting of the node labelled with fact in θ(Q) and its ancestors in the chase graph $\mathcal{G}(D, \Sigma)$. In such tree, denoted as $\mathcal{T}(\theta(\mathrm{Q}))$, there are at least two facts $\mathbf{f_1}$ and $\mathbf{f_2}$ that are equivalent up renaming labelled nulls (i.e., isomorphic) and contribute to a positive answer to Q. By [4](Th. 2), the $subgraph(\mathcal{G}, \mathbf{f_1})$ is equivalent up to renaming labelled nulls to $subgraph(\mathcal{G}, \mathbf{f_2})$ (i.e., are isomorphic), namely, there exist two homomorphisms Δ_1 and Δ_2 such that $\Delta_1(subgraph(\mathcal{G}, \mathbf{f_1})) = subgraph(\mathcal{G}, \mathbf{f_2})$ and $\Delta_2(subgraph(\mathcal{G}, \mathbf{f_2})) = subgraph(\mathcal{G}, \mathbf{f_1})$. Now we distinguish two cases:

(1) $\underline{\mathbf{f_2} \in subgraph(\mathcal{G}, \mathbf{f_1})}$ (resp. $\mathbf{f_1} \in subgraph(\mathcal{G}, \mathbf{f_2})$). In this case, as both $\mathbf{f_1}$ and $\mathbf{f_2}$ contribute to answer Q, $\theta(Q) \in subgraph(\mathcal{G}, \mathbf{f_2})$ (resp. $\theta(Q) \in subgraph(\mathcal{G}, \mathbf{f_1})$). We can construct the homomorphism $\theta' = \theta \circ \Delta_2$ (resp. $\theta' = \theta \circ \Delta_1$), such that $\theta'(Q) \in subgraph(\mathcal{G}, \mathbf{f_1})$ (resp. $\theta'(Q) \in subgraph(\mathcal{G}, \mathbf{f_2})$). This contradicts the fact that there exist two isomorphic facts.
(2) $\underline{\mathbf{f_1} \notin subgraph(\mathcal{G}, \mathbf{f_2}) \text{ and } \mathbf{f_2} \notin subgraph(\mathcal{G}, \mathbf{f_1})}$. Consider the sets of facts $M = \mathcal{T}(\theta(\mathrm{Q})) \cap subgraph(\mathcal{G}, \mathbf{f_1})$ and $M' = \mathcal{T}(\theta(\mathrm{Q})) \cap subgraph(\mathcal{G}, \mathbf{f_2})$. Such sets correspond to the ancestors of the fact $\theta(Q)$ in the subgraphs of $\mathbf{f_1}$ and $\mathbf{f_2}$. As the subgraphs are isomorphic, even such sets are isomorphic. Moreover, $M \cap M' \neq \emptyset$ as at least contains $\theta(Q)$. Consider the possible pairs of non-equivalent facts (p_1, p_2) such that $p_1 \in M$, $p_2 \in M'$, $\Delta_1(p_1) = p_2$ and $\Delta_2(p_2) = p_1$. Observe that we can replace each fact p_2 with p_1 (resp. p_1 with p_2) for each TGD chase step application and still obtain an equivalent number of derivations and the fact $\theta(Q)$. In fact, (i) p_1 and p_2 are equivalent up to renaming labelled nulls (they refer to the same predicate name and they have the same constants in the same positions), hence they activate the same TGD chase steps; (ii) by wardedness, each fact inherits its labelled nulls from exactly one parent [4] and this preserves the identity of the labelled nulls in $\theta(Q)$; (iii) by the absence of harmful joins, the labelled nulls identity is irrelevant to create the facts in M (resp. M'), i.e., to apply any TGD chase step. Again, this contradicts the fact that there exist two isomorphic facts.

This theorem can be extended to BCQs. Indeed, a BCQ $Q : \boldsymbol{\phi}(\mathbf{x}, \mathbf{y})$ can be interpreted as a warded TGD with a head of 0-arity and thus can be inserted in Σ_T. Therefore, after harmful joins removal [3], answering Q is equivalent to answering an atomic query. □

8 EXISTENCE AND FINITENESS OF $chase^B(D, \Sigma_T)$

Theorem 3.6a. *Given a set of warded TGDs Σ_T and a database D, the universal model for $D \cup \Sigma_T$ $chase^B$ is finite.*

Proof (sketch). Consider a BCQ Q and a homomorphism θ such that $\theta(Q) \subseteq chase(D, \Sigma_T)$. The finiteness of $chase^B(D, \Sigma_T)$ can be proven as a purely combinatorial computation of the maximum derivation level of a fact satisfied by θ in the chase graph $\mathcal{G}$ for D and Σ_T. It is sufficient to show that there exists always a constant δ, which depends

only on $\mathbf{S}$, Σ_T and Q, such that Q can always be answered on $chase^{\delta}(D, \Sigma)$, i.e., the initial and finite portion $\mathcal{G}$ until derivation level δ. Then we can assume $chase^{B}(D, \Sigma_T) = chase^{\delta}(D, \Sigma_T)$.

We first show that there exists a constant $\mathbf{k}$ such that every set $\mathbf{M}$, containing facts of the same predicate in $\mathbf{S}$, contains at least two isomorphic facts if its size is greater than $\mathbf{k}$, i.e., $|\mathbf{M}| > \mathbf{k}$. We compute an upper bound for $\mathbf{k}$ as follows. Let $\mathbf{w}$ be the maximal arity of a predicate in $\mathbf{S}$. Let $dom(D)$ be the set of constants in the database D. In the worst case, about $(dom(D) + \mathbf{w})^{\mathbf{w}}$ non-isomorphic facts can be created using constants in $dom(D)$ and new labelled nulls. Indeed, $dom(D)^{\mathbf{w}}$ is the number of possible facts (that refer to the same predicate) created with constants in $dom(D)$. Then we derive the quantity $dom(D) + \mathbf{w}$ by considering also disposals for new labelled nulls (at most $\mathbf{w}$), whose identity is irrelevant to create new facts by wardedness. Now, assume $\mathbf{k} = (dom(D) + \mathbf{w})^{\mathbf{w}}$. If we consider $\mathcal{G}$ until derivation level $|\Sigma_T| \cdot \mathbf{k}$, all the possible non-isomorphic facts are generated in the chase. In fact, in the worst case, at least a a fact referring to a certain predicate is generated by activating all the TGDs in Σ_T. Furthermore, by assuming derivation level $\delta = |Q| \cdot |\Sigma_T| \cdot \mathbf{k}$, we cover also the maximum number of isomorphic copies required to answer a specific query Q. Hence, $chase^{\delta}(D, \Sigma_T) \models Q$ and we conclude that $chase^{B}(D, \Sigma_T)$ is finite by assuming $chase^{B}(D, \Sigma_T) = chase^{\delta}(D, \Sigma_T)$. □

9 DECIDABILITY OF BCQ QUERY ANSWERING UNDER WARDED TGDS AND HARMLESS EGDS

Consider the theoretical chase $chase^{B}(D, \Sigma_T) \subseteq chase(D, \Sigma_T)$, finite and BCQ-equivalent to $chase(D, \Sigma_T)$ i.e., a universal model for $D \cup \Sigma_T$. Definition 3.2 suggests an approach to decide BCQs in the presence of a set of dependencies $\Sigma = \Sigma_T \cup \Sigma_E$, where Σ_E is a set of harmless EGDs with respect to Σ. Such procedure, can be summarized in two distinct steps.

(1) **Satisfiability**. Decide weather $D \cup \Sigma$ is satisfiable; if not, conclude $D \cup \Sigma \models Q$. The only way $D \cup \Sigma$ is not satisfiable is a hard violation of an EGD, i.e., equating two constants. Intuitively, the main idea behind our technique to check satisfiability is to separate the effects of EGDs into (i) hard violations and (ii) soft violations, i.e., equating labelled nulls with other values.
(2) **Homomorphism**. If $D \cup \Sigma$ is satisfiable, there exists a homomorphism h mapping $chase(D, \Sigma_T)$ onto $chase(D, \Sigma)$ such that $chase(D, \Sigma) \models Q$ iff $h(chase(D, \Sigma_T)) \models Q$. We can determine h and check whether $h(chase(D, \Sigma_T)) \models Q$. A technique to build a homomorphism from $chase(D, \Sigma_T)$ onto $chase(D, \Sigma)$ is as follows. We initialize an empty homomorphism $h = \emptyset$. Then we apply the EGDs of Σ_E on $chase^{B}(D, \Sigma_T)$ to fixpoint, i.e., we compute $chase^{H}(D, \Sigma) = chase(chase^{B}(D, \Sigma_T), \Sigma_E)$.

The next result argues for the overall correctness of our procedure.

Theorem 3.6b. *Given a set $\Sigma = \Sigma_T \cup \Sigma_E$ of warded TGDs and harmless EGDs with respect to Σ, and a database D, for every BCQ Q, is decidable to check whether $chase(D, \Sigma) \models Q$.*

Proof. We prove the two steps of the procedure.

(1) Checking Satisfiability. The only way $D \cup \Sigma$ is not satisfiable is a hard violation of an EGD, i.e., equating two constants. Intuitively, the main idea behind our technique to check satisfiability is to separate the effects of EGDs into (i) hard violations and (ii) soft violations, i.e., equating labelled nulls with other values (nulls or constants). To repair soft violations, we rewrite each EGD η into a pair of EGDs η' and η'', where we alternatively force x_i and x_j to bind to non-ground values. Hard violations, and thus satisfiability, are then detected iff none of the queries Q_V of a set $\mathbf{Q}_V$ of check queries for hard violations hold.
The encoding is done as follows. We first construct sets D' and Σ'_E, which will make up $D' \cup \{\Sigma_T \cup \Sigma'_E\}$ against which we will perform our queries. We extend D into D' by adding facts $neq(c_1, c_2)$ for each pair of distinct constants

$c_1, c_2 \in dom(D)$. We define a new set of EGDs Σ'_E, containing two EGDs, η' and η'', for each $\eta \in \Sigma_E$, built as follows. For each EGD $\eta : \boldsymbol{\phi}(\mathbf{x}) \rightarrow x_i = x_j$ of Σ_E, we let: $\eta' : \boldsymbol{\phi}(\mathbf{x}), null(x_i) \rightarrow x_i = x_j$, and $\eta'' : \boldsymbol{\phi}(\mathbf{x}), null(x_j) \rightarrow x_i = x_j$. The artificial *null* predicate forces x_i (resp. x_j) to bind only to labelled nulls, i.e., $\vDash null(x)$ iff $x \in \mathbf{N}$.
It is easy to check that assuming $D \cup \Sigma$ is satisfiable, $D \cup \Sigma_T$ is logically equivalent to $D \cup \{\Sigma_T \cup \Sigma'_E\}$, as the only failing cases arise when both x_i and x_j are bound to constants in some EGD and $x_i \neq x_j$ (i.e., hard violations). On the other hand, such EGD activations produce no effects if $x_i = x_j$. We now construct the set of queries $\mathbf{Q}_V$ that check for hard violations. For each $\eta \in \Sigma_E$, we add to $\mathbf{Q}_V$ the following BCQ: $q \leftarrow \boldsymbol{\phi}(\mathbf{x}), neq(x_i, x_j)$. We have that $D' \cup \{\Sigma_T \cup \Sigma'_E\}$ is always satisfiable, since there cannot be hard violations by construction. Moreover, as Σ_E is harmless, Σ'_E is harmless as well.

(2) Finding the Homomorphism. A technique to build a homomorphism from $chase(D, \Sigma_T)$ onto $chase(D, \Sigma)$ is as follows. We initialize an empty homomorphism $h = \emptyset$. We then apply the EGDs of Σ_E on $chase^B(D, \Sigma_T)$ to fixpoint, i.e., we compute $chase^H(D, \Sigma) = chase(chase^B(D, \Sigma_T), \Sigma_E)$. This is finite for a fixed set of EGDs [5]. For each EGD chase step $\boldsymbol{\varphi}(\mathbf{x}) \xrightarrow{\eta_\theta} x_i = x_j$ that binds a labelled null $\theta(x_i)$ to either a constant or another labelled null $\theta(x_j)$, we update h by adding the assignment $\theta(x_i) \rightarrow \theta(x_j)$, and replacing all occurrences of $\theta(x_i)$ appearing in the right-hand side of any assignment in h with $\theta(x_j)$. Then, for each labelled null ν' appearing in a fact $\boldsymbol{\phi'}(\mathbf{x})$ of $chase(D, \Sigma_T) \setminus chase^B(D, \Sigma_T)$, let ν be the labelled null appearing in an equivalent fact, up to the renaming of $\nu \rightarrow \nu'$, in $\boldsymbol{\phi}(\mathbf{x})$ of $chase^B(D, \Sigma)$, which exists by universality of $chase^B$. If h contains a mapping from ν to a term μ, we extend it by adding the assignment $\nu' \rightarrow \mu$. Then, as $h(chase(D, \Sigma_T))$ is finite by construction, $D \cup \Sigma \models Q$ can be decided by checking whether $h(chase(D, \Sigma_T)) \models Q$.

□

10 PROOF OF THEOREM 3.7

Theorem 3.7. *Given a set $\Sigma = \Sigma_T \cup \Sigma_E$ of warded TGDs and harmless EGDs with respect to Σ, and a database D, if $D \cup \Sigma$ is satisfiable, for every BCQ Q, it holds $chase^H(D, \Sigma) \models Q$ iff $chase(D, \Sigma) \models Q$.*

Proof. We prove the two directions of the implication separately.

($\Rightarrow$) Let us proceed by contradiction and assume there exists a BCQ $Q : q \leftarrow \boldsymbol{\phi}(\mathbf{x})$ such that $chase^H(D, \Sigma) \models Q$ and $chase(D, \Sigma) \not\models Q$. Thus, there exists a homomorphism θ such that $\theta(\boldsymbol{\phi}(\mathbf{x})) \in chase^H(D, \Sigma)$ and there does not exist any homomorphism θ' such that $\theta'(\boldsymbol{\phi}(\mathbf{x})) \in chase(D, \Sigma)$. By harmlessness of Σ_E, we have that for a homomorphism h, it holds $h(chase(D, \Sigma_T)) \models Q$ iff $chase(D, \Sigma) \models Q$. Therefore, there does not exist any fact (or conjunction thereof) $\boldsymbol{\varphi}(\mathbf{x}) \in chase(D, \Sigma_T)$ such that $h(\boldsymbol{\varphi}(\mathbf{x})) \in chase(D, \Sigma)$, with $\theta'(\boldsymbol{\phi}(\mathbf{x})) = \boldsymbol{\varphi}(\mathbf{x})$. Then, since Σ_T is warded, there does not exist any fact $\boldsymbol{\varphi'}(\mathbf{x}) \in chase^B(D, \Sigma^T)$, where $\boldsymbol{\varphi'}(\mathbf{x}) = \boldsymbol{\varphi}(\mathbf{x})$ up to renaming of labelled nulls and such that $h(\boldsymbol{\varphi'}(\mathbf{x})) \in chase(D, \Sigma)$. Yet, we have that $\theta(\boldsymbol{\phi}(\mathbf{x})) \in chase^H(D, \Sigma)$ and so, since Σ_E is harmless and h is surjective, there is some fact $\boldsymbol{\varphi'}(\mathbf{x}) \in chase^B(D, \Sigma_T)$ such that $\theta(\boldsymbol{\phi}(\mathbf{x})) = h(\boldsymbol{\varphi'}(\mathbf{x}))$, which we assumed not to exist. Having reached a contradiction, we conclude $chase(D, \Sigma) \models Q$.

($\Leftarrow$) Let us proceed again by contradiction and assume there exists a BCQ $q \leftarrow \boldsymbol{\phi}(\mathbf{x})$ such that $chase(D, \Sigma) \models Q$ and $chase^H(D, \Sigma) \not\models Q$. Thus, there exists a homomorphism θ such that $\theta(\boldsymbol{\phi}(\mathbf{x})) \in chase(D, \Sigma)$ and there does not exist any homomorphism θ' such that $\theta'(\boldsymbol{\phi}(\mathbf{x})) \in chase^H(D, \Sigma)$. By harmlessness of Σ_E, we have that for a homomorphism h, it holds $h(chase(D, \Sigma_T)) \models Q$ iff $chase(D, \Sigma) \models Q$. Then there exists a fact (or conjunction thereof) $\boldsymbol{\varphi}(\mathbf{x}) \in chase(D, \Sigma_T)$ such that $h(\boldsymbol{\varphi}(\mathbf{x})) = \theta(\boldsymbol{\phi}(\mathbf{x})) \in chase(D, \Sigma)$. Now, since Σ_T is warded, we have that $\boldsymbol{\varphi}(\mathbf{x}) \in chase^B(D, \Sigma)$, modulo fact isomorphism. However, since we have assumed that there does not exist a homomorphism θ' such that $\theta'(\boldsymbol{\phi}(\mathbf{x})) \in$

$chase^H(D,\Sigma)$, by harmlessness of Σ_E and wardedness of Σ_T, there cannot exist any fact $\theta'(\boldsymbol{\phi}(\mathbf{x})) = \boldsymbol{\varphi}(\mathbf{x}) \in chase^B(D,\Sigma)$ such that $\boldsymbol{\varphi}(\mathbf{x}) \in chase^H(D,\Sigma)$. Having reached a contradiction, we conclude $chase^H(D,\Sigma) \models Q$. □

11 PROOF OF THEOREM 3.8

So far we have followed a pragmatic approach, augmenting the universal semantics of $D \cup \Sigma_T$ by chasing it under Σ_E: this idea lends itself to a practical application, once an efficient implementation for $chase^B$ is provided. However, it does not allow us to draw conclusions about the data complexity of the new fragment. In fact, the PTIME membership proof [8] of Warded Datalog$^\pm$ hinges on the possibility to generate the ground semantics $\Sigma(D)_\downarrow = \{\overline{a} : \overline{a} \in chase(D,\Sigma) \wedge dom(a) = \mathbf{C}\}$ in polynomial time and does not consider the universal one. Hence, it is to prove that the presence of harmless EGDs does not increase the complexity of computing the ground semantics. Our proof (in the Appendix) extends the membership proof for the sole warded fragment. The differences that make our task challenging are the need for a satisfiability check and a refocus of the PTIME membership proof of $\Sigma(D)_\downarrow$ for harmless EGDs.

The following results first show time complexity with warded TGDs and harmless EGDs, under the assumption $D \cup \Sigma$ is satisfiable (Theorem 3.8). Then, we illustrate how the satisfiability problem can be reduced to an instance of CQ answering over warded TGDs and harmless EGDs, where $D \cup \Sigma$ is satisfiable (Theorem 3.9).

Theorem 3.8. *BCQ answering for Warded Datalog$^\pm$ and harmless EGDs is PTIME-complete in data complexity*

Proof. The lower bound descends from Datalog being PTIME-hard [7]. It is easy to verify that any set of Datalog dependencies are also a set of Warded Datalog$^\pm$ dependencies: in the absence of existential quantification, there are no affected variables and the warded condition is trivially respected. So we concentrate on the upper bound, considering a query Q, a database D and a fixed set $\Sigma = \Sigma_T \cup \Sigma_E$ of warded TGDs and harmless EGDs. Our goal is proving that deciding whether $D \cup \Sigma \models Q$ is feasible in polynomial time in D.

Since we assume that $D \cup \Sigma$ is satisfiable, we can proceed with the following two steps. After a preliminary elimination of the negation (step (1)), we just prove that checking whether an atom $\Psi(\mathbf{t})$ belongs to the ground semantics $\Sigma^+(D^+)_\downarrow$ of the (negation-free) Σ^+ can be done in polynomial time (step (2)). The correctness of the proof approach is straightforward.

- By construction, we have that $D^+ \cup \Sigma^+ \models Q$ if for some $\Psi(\mathbf{t}) \in \Sigma^+(D^+)_\downarrow$, it holds that $\Psi(\mathbf{t}) \in Q(\Sigma^+(D^+)_\downarrow)$. In other terms, to check whether Q is satisfied, we need to scan the ground semantics $\Sigma^+(D^+)_\downarrow$.
- The size of $\Sigma^+(D^+)_\downarrow$ is polynomial in the size of D: the result has been proven for warded TGDs [2] and is not affected by harmless EGDs, which cannot in turn trigger TGDs and produce new facts.

(1) Elimination of Negation: We build a database $D^+ \supseteq D$ and eliminate the negation from Σ and Q, so that $D^+ \cup \Sigma^+ \models Q$ iff $D \cup \Sigma \models Q$. The procedure to build D^+ and Σ^+ is the standard one [9], which applies to EGDs as well, and so $\Sigma^+ = \Sigma_T^+ \cup \Sigma_E^+$. Since the negation in Warded Datalog$^\pm$ is stratified and ground, Σ^+ is computed from Σ by iteratively replacing each negative atom $\neg a(\mathbf{x})$ with a positive atom $\overline{a}(\mathbf{x})$, where the extension of $\overline{a}(\mathbf{x})$ in D^+ is the complement of a with respect to the ground semantics $\Sigma(D)_\downarrow$ (see step (2)).

(2) Building the Ground Semantics: It remains to show the crucial technical lemma that the negation-free —and we shall omit the plus superscript from hereinafter— $\Sigma(D)_\downarrow$ can be built in polynomial time. As we have argued, this task has the same complexity as checking whether a fact $\Psi(\mathbf{t})$ belongs to $\Sigma^+(D^+)_\downarrow$.

Let us first normalize Σ_T so that every TGD σ having more than one body atom is either *head-ground*, i.e., all the head terms are constants, harmless and universally quantified variables, or *semi-body-ground*, i.e., there exists at most one atom in $body(\sigma)$ containing a harmful variable. We partition Σ_T into $\{\Sigma_h, \Sigma_b\}$, where Σ_h is the set comprising

all the head-ground TGDs, Σ_b is the set of semi-body-ground TGDs. Without loss of generality, we will consider the presence of at most one existentially quantified variable in the TGDs.

We now provide a high-level description of a procedure EgdProofTree($D, \Sigma_h, \Sigma_b, \Sigma_E, \Psi(\mathbf{t})$) that takes as input a database D, a set of warded TGDs partitioned into $\{\Sigma_h, \Sigma_b\}$ as described, a set of harmless EGDs Σ_E, and a fact $\Psi(\mathbf{t})$. The procedure accepts if $\Psi(\mathbf{t}) \in \Sigma(D)_{\downarrow}$, rejects otherwise. The procedure builds a proof tree. Roughly speaking, a proof tree can be obtained from a proof (an execution of the chase, or a chase graph), by reversing the edges and unfolding the obtained graphs into a tree, by repeating some nodes. Conversely, we can obtain a chase graph, by reversing the edges of a proof tree and collapsing some nodes. Intuitively, building the two structures is equivalent. We apply resolution steps starting from $\Psi(\mathbf{t})$ until the database D is reached. If not, $\Psi(\mathbf{t})$ is rejected. This is done via an alternating procedure that builds the various branches of a proof tree for $\Psi(\mathbf{t})$ in parallel universal computations, ensuring the compatibility of the branches. This is the main technical challenge, as we need to ensure that the introduced labelled nulls occurring in different branches of the proof tree represent the same term of $\Sigma(D)_{\downarrow}$, even when harmless EGDs are involved.

Let us introduce some conceptual tools we will use in the procedure.

- the *taint cause* $\chi(p[i])$ for a tainted position $p[i]$ is the set of EGDs of Σ_E that can assign the value of a labelled null in $p[i]$. More formally, $\chi(p[i])$ can be inductively defined as the set of all the EGDs $\eta : \boldsymbol{\phi}(\mathbf{x}) \rightarrow x_i = x_j$ of Σ_E such that: (i) $p[i]$ is the position of x_i (resp x_j) in $body(\eta)$ and $p[i]$ is harmful in η, or, (ii) for some TGD $\sigma \in \Sigma_T$ and some variable $v \in body(\sigma) \cap head(\sigma)$, we have that v appears in a body (resp. head) position that is in $\chi(p[i])$ and in position $p[i]$ in $head(\sigma)$ (resp. $body(\sigma)$). For the sake of simplicity, in the rest of the proof we will assume that x_i is the harmful variable in the EGD body.
- Given an atom Ψ, a TGD $\sigma \in \Sigma_h$, and a homomorphism θ from $head(\sigma)$ to Ψ, a *head-ground TGD resolution step* $\Psi \triangleright^h_{\theta_\sigma}$ produces the set of atoms $\{\theta^B_\sigma(b) : b \in body(\sigma)\}$, where θ^B_σ is the extension of θ that assigns variables appearing $body(\sigma) \setminus head(\sigma)$ to fresh labelled nulls or constants from $dom(D)$.
- Given an atom Ψ, a TGD $\sigma \in \Sigma_h$, and a homomorphism θ from $head(\sigma)$ to Ψ, a *semi-body-ground TGD resolution step* $\Psi \triangleright^h_{\theta_\sigma}$ produces the set of atoms $\{\theta^B_\sigma(b) : b \in body(\sigma)\}$, where θ^B_σ is the extension of θ that assigns variables appearing $body(\sigma) \setminus head(\sigma)$ in other atoms than the ward, to constants from $dom(D)$. The variables in the ward are mapped to either labelled nulls or constants from $dom(D)$.
- Given an atom $\Psi(\mathbf{t})$ where $\pi = \Psi(t[i])$ is a tainted position, an EGD $\eta : \boldsymbol{\phi}(\mathbf{x}) \rightarrow x_i = x_j$, with $\eta \in \chi(\pi)$, a homomorphism θ that maps x_j to $\Psi(t[i])$, a *an EGD resolution step* $\Psi(\mathbf{t}) \triangleright^{\pi}_{\theta_\eta}$ produces the set of atoms $\{\theta^B_\eta(b) : b \in body(\eta)\} \cup \{\Psi'(\mathbf{t})\})\}$, where θ^B_η is the extension of θ that assigns variables appearing $body(\sigma)$ to either fresh labelled nulls or constants from $dom(D)$, and $\Psi'(\mathbf{t})$ is obtained from $\Psi(\mathbf{t})$, by replacing the value at $\Psi(t[i])$ with $\theta^B_\eta(x_i)$. The intuition behind a partial EGD resolution step is explaining the value $\Psi(t[i])$ of one single tainted position via one EGD application.
- Given a set of atoms S, and a set of labelled nulls $N \subseteq \mathbf{N}$, a partition $\Pi_N(S)$ is known as as an *[N]-linking* partition [1] if $\Pi_N(S)$ is a partition for S and for each labelled null $\nu \in (dom(S) \cap \mathbf{N}) \setminus N$ there is exactly one subset $S_i \in \Pi_N(S)$ in which ν appears. A partition $\Pi_N(S)$ is defined as *[N]-optimal* if it is [N]-linking and there does not exist any other [N]-linking partition $\Pi'_N(S)$ such that $|\Pi'_N(S)| > |\Pi_N(S)|$. For example, consider the set $S = \{p(c, z_1), p(z_1, z_2), p(z_2, z_3), p(z_3, z_4)\}$, where $z_1, z_2, z_3, z_4 \in \mathbf{N}$ and let $N = \{z_2, z_3, z_4\}$. The partition $\{\{p(c, z_1), p(z_1, z_2)\}, \{p(z_2, z_3), p(z_3, z_4)\}\}$ is [N]-linking as every $z_i \in dom(S) \cap \mathbf{N}$ occurs exactly in one component. Yet, it is not [N]-optimal, because the partition $\{\{p(c, z_1), p(z_1, z_2)\}, \{p(z_2, z_3)\}, \{p(z_3, z_4)\}\}$ is

still [N]-linking, but has higher cardinality. It turns out, the latter is [N]-optimal, as $\{p(c, z_1), p(z_1, z_2)\}$ cannot be split without separating the z_1 occurrences.

The procedure performs the following steps. Let us first initialize $S = \{\Psi(\mathbf{t})\}$. Start from step (a).

(1) **EGD resolution**. For each $\Psi(\mathbf{t}) \in S$, guess whether skipping to step (b). If not, if $\Psi(\mathbf{t})$ does not contain any tainted position, go to step (b). If $\Psi(\mathbf{t})$ contains a tainted position π, guess an EGD $\eta \subseteq \Sigma_E$ such that $\eta \in \chi(\pi)$ and let θ_η be a homomorphism from x_j to $\Psi(\mathbf{t}[\pi])$. Apply the resolution step $\Psi(\mathbf{t}) \triangleright^{\pi}_{\theta_\eta} \{b_{\Psi_1}, \ldots, b_{\Psi_n}\}$ and let $S_\Psi = \{b_{\Psi_1}, \ldots, b_{\Psi_n}\}$ be the generated atoms. Then, with $S^+ = \bigcup_{\Psi \in S} S_\Psi$, we compute $\Pi_N(S^+) = \{S^+_1, \ldots, S^+_n\}$ as the $[\emptyset]$-optimal partition of S^+. Universally select and prove in parallel computations each element of $\Pi_\emptyset(S^+)$, recursively from step (a). The goal of this partitioning strategy is keeping in the same universal computation all the labelled nulls appearing in more than one branch.

(2) **TGD $\in \Sigma_h$ resolution**. If $\Psi(\mathbf{t}) \in D$ then accept. If $dom(\Psi(\mathbf{t})) \cap \mathbf{N} \neq \emptyset$, then go to step (c) with $\{\Psi(\mathbf{t})\}$. Else, guess a TGD $\sigma \in \Sigma_h$ and a homomorphism θ_σ from $head(\sigma)$ to $\Psi(\mathbf{t})$. Apply the resolution step $\Psi(\mathbf{t}) \triangleright^{h}_{\theta_\sigma} S$, where $S = \{b_1, \ldots, b_n\}$ are the generated atoms. If such σ or θ_σ cannot be found, then reject. Let $\Pi_\emptyset(S)$ be an $[\emptyset]$-optimal partition of S. Universally select each $S \in \Pi_\emptyset(S)$ and proceed as follows: if $S = \{\Psi'(\mathbf{t}')\}$ and $dom(S_i) \subseteq dom(D)$, recursively call step (b) for $\Psi'(\mathbf{t}')$; else, that is, if $dom(S) \cap \mathbf{N} \neq \emptyset$, go to step (c).

(3) **TGD $\in \Sigma_b$ resolution**. Let us initially mark all the labelled nulls occurring in S, by storing them into a set $R_S = \{(v, \epsilon) : v \in dom(S) \cap \mathbf{N}\}$. For each $a \in S$ guess a rule $\sigma \in \Sigma_b$ and a homomorphism θ_σ from $head(\sigma)$ to $\Psi(\mathbf{t})$. If, for any a, the labelled null $\overline{v}$ occurs in an existentially quantified position of σ, then mark it by updating the corresponding element in R_s as $(\overline{v}, a)$. Apply the resolution step $\Psi(\mathbf{t}) \triangleright^{b}_{\theta_\sigma} S_a$, where $S_a = \{b_{a_1}, \ldots, b_{a_m}\}$ are the generated atoms. If such σ or θ_σ cannot be found, then reject. Then, with $S^+ = \bigcup_{a \in S} S_a$, we let $N = \{v \in dom(S^+) \cap \mathbf{N} : (v, x) \in R_s, x \neq \epsilon\}$ and compute $\Pi_N(S^+) = \{S^+_1, \ldots, S^+_n\}$ as the $[N]$-optimal partition of S^+. Intuitively, we are separating labelled nulls for which a generating atom, via its existential quantifier, has been found, while we are keeping the others linked. In this way, it is sufficient to store the generating atom along with the invented labelled nulls, to be able to safely merge different branches. Then for each $S_i \in \Pi_N(S^+)$, the new labelled nulls in S^+_i are marked, we universally select $S \in \{S^+_1, \ldots, S^+_n\}$ and proceed as follows: if $S = \{\Psi'(\mathbf{t}')\}$ and $dom(S_i) \subseteq dom(D)$, recursively call step (b) for $\Psi'(\mathbf{t}')$; else, that is, if $dom(S) \cap \mathbf{N} \neq \emptyset$, go to step (c).

The correctness of the procedure follows by construction and harmlessness: if existing, an accepting computation for $\Psi(\mathbf{t})$ can be found by first resolving all the EGDs as they do not affect the applicability of TGDs and then, once step (a) non-deterministically skips to (b), by resolving the TGDs (by applying the procedure from [1]).

Let us briefly discuss the labelled null life cycle throughout the procedure. A fresh labelled null can be invented in resolution steps in different cases. An EGD step can introduce at most one labelled null for each variable in the body other than x_j; it can also introduce one more labelled null as a placeholder in the tainted position of Ψ'. A TGD step introduces nulls whenever an atom is resolved with a rule having body variables not appearing in the head. Nulls then propagate with resolution steps, copied from the rule heads into the resolving atoms. A null is suppressed when it is resolved via an existentially quantified variable in a rule head. Clearly, the overall assignment of the nulls must be coherent in all the universal computations. Therefore it is essential that the atoms where the null v is invented in different parallel computations are isomorphic to $\theta(head(\sigma))$. To ensure that, the partitioning technique keeps together within the partition that contains v, the atom $\theta(head(\sigma))$. In this way, the different branches can be eventually merged in a sound way, with a global coherence of the nulls.

Space Complexity of the Steps. We encode EGDProofTree$(D, \Sigma_h, \Sigma_b, \Sigma_E, \Psi(\mathbf{t}))$ as EGDOnlyProofTree$(D, \Sigma_E, \Psi(\mathbf{t}))$ $\bigcup_i$ TGDProofTree$(\mathbf{a_i}, \Sigma_h, \Sigma_b, \Psi(\mathbf{t}))$, that is, a partial proof tree built by applying only steps (a) followed by multiple proof trees built with the application of steps (b) and (c) on all the leaves $\mathbf{a_i}$ of the first proof tree. By harmlessness of Σ_E, this decomposition is correct as when a computation of (a) non-deterministically skips to (b), no application of step (a) will be performed again. Intuitively, once the EGDs have been used to explain $\Psi(\mathbf{t})$ by a set of TGD outputs, only TGDs are used in the following resolution steps, coherently with harmlessness. It has been proven that a single TGDProofTree$(D, \Sigma_h, \sigma_b, \mathbf{a_i})$ can be generated in polynomial time, by showing that steps (b) and (c) use $O(log(dom(D)))$ space at each step and remembering that that alternating logarithmic space (ALOGSPACE) coincides with PTIME [1]. It remains to show that the application of steps (a) requires polynomial time.

We start by proving that the size of each component of a $[\emptyset]$-optimal partition of S^+ in EGDOnlyProofTree is at most $AB(B+1)$, where B is $max_{\eta \in \Sigma_E}|body(\eta)|$ and A is the maximum arity of an atom of $body(\eta)$. Intuitively, each step adds at most $B+1$ new atoms, until all the possibly tainted variables (A) of all the body atoms (B) have been resolved. At that point, each component of $\Pi_\emptyset(S_i^+)$ stably contains at most $AB(B+1)$ elements by construction. Each subsequent steps removes and re-adds at most $B+1$ atoms.

More formally, let us proceed by induction on the number of applied steps (a). As the base case, we have that the first partitioning step produces $\Pi_\emptyset(S_0^+) = \{S^0_{\Psi_1}, \ldots, S^0_{\Psi_n}\}$. Each $S^0_{\Psi_i} \in \Pi_\emptyset(S^+)$ contains at most $B+1$ elements, as in the worst case, we resolve a (ground) tainted position of $\Psi(\mathbf{t})$ via an EGD resolution step that produces the maximum number of atoms of $body(\eta)$, all sharing a fresh labelled null, plus one atom $\Psi'(\mathbf{t})$, sharing the labelled null $\theta^B_\eta(x_i)$ used to replace the tainted position with one atom in $body(\eta)$. In the inductive case, we consider a component $S^k_{\Psi_i} \in \Pi_\emptyset(S^+_k)$ generated during the k-th step, with $k > 1$ by resolving the atoms $\mathbf{a}$ in $S^{k-1}_{\Psi_j} \in \Pi_\emptyset(S^+_{k-1})$. By inductive hypothesis we assume $|S^{k-1}_{\Psi_j}| \leq AB(B+1)$. Two cases are possible for each resolution step $\Psi(\mathbf{t}) \overset{\pi}{\triangleright}_{\theta_\eta} \{b_{\Psi_1}, \ldots, b_{\Psi_n}\}$ applied on an atom $\mathbf{a}^*$ before the partitioning step k: (i) the variable $\mathbf{a}^*[\pi]$ is ground; (ii) $\mathbf{a}^*[\pi]$ is a labelled null ν introduced in a previous resolution step by applying $\theta^B_\eta(x_i)$, and π is a tainted position for $body(\eta)$. In case (i), as π is ground, no resolution step has ever been applied on that tainted position; hence, at least one EGD η has never been applied in a resolution step. By applying the inductive hypothesis it follows that $S^{k-1}_{\Psi_j}$ contains at most $(B+1)(AB-1)$ atoms, and $|S^k_{\Psi_i}| \leq AB(B+1)$. In case (ii), the resolution step generates at most $B+1$ new atoms in the same partition, including the atom $\mathbf{a}^{*\prime}$ obtained by replacing the tainted variable x_i of $\mathbf{a}^*$ in position π, with $\theta^B_\eta(x_i)$. The introduction of a fresh labelled null in $\mathbf{a}^{*\prime}[\pi]$, disconnects this atom from at least $B+1$ other atoms potentially in the same partition, whence $|S^k_{\Psi_i}| \leq |S^{k-1}_{\Psi_j}| \leq AB(B+1)$.

Having set a bound on the size of the $[\emptyset]$-optimal components of S^+, it is easy to see that we need to remember at most $(AB(B+1))^2$ atoms, at each step. The space needed to represent such atoms is polynomial in Σ_E and logarithmic in $|dom(D)|$. If Σ_E is fixed, it follows that EGDOnlyProofTree$(D, \Sigma_E, \Psi(\mathbf{t}))$ uses $O(log|dom(D)|)$ space at each step of its computation, and thus the proof tree can be generated in PTIME. In total, remembering that ALOGSPACE coincides with PTIME, we conclude that EGDOnlyProofTree and so EGDProofTree can be generated in polynomial time. □

We now show how the satisfiability problem can be reduced to CQ answering over warded TGDs and harmless EGDs where $D \cup \Sigma$ is satisfiable and provide the full complexity result.

Theorem 3.9. *CQ answering for Warded Datalog$^\pm$ and harmless EGDs is PTIME-complete in data complexity.*

Proof. Let Q be a query, D be a database, and $\Sigma = \Sigma_T \cup \Sigma_E$ a fixed set of warded TGDs and harmless EGDs. We proceed by case distinction based on whether $D \cup \Sigma$ is satisfiable. If $D \cup \Sigma$ is satisfiable, by Lemma 3.8, we can conclude

that the problem of deciding $D \cup \Sigma \models Q$ is PTIME in data complexity. It remains to show that deciding on the satisfiability of $D \cup \Sigma$ can be done in PTIME.

We first give a sketch of the proof, and then describe the construction. Observe that the only way $D \cup \Sigma$ is not satisfiable is a hard violation of an EGD, i.e., equating two constants. Intuitively, the main idea behind this proof is to separate the effects of EGDs into (i) hard violations, i.e., equating constants, and (ii) soft violations, i.e., equating labelled nulls with other values (nulls or constants). To detect and correct soft violations, we rewrite each EGD η into a pair of EGDs η' and η'', where we alternatively force x_i and x_j to bind to non-ground values. Hard violations, and thus satisfiability, are then detected iff none of the queries Q_V of a set $\mathbf{Q}_V$ of check queries for hard violations hold. The encoding is done as follows.

We first construct sets D' and Σ'_E, which will make up $D' \cup \{\Sigma_T \cup \Sigma'_E\}$ against which we will perform our queries. We extend D into D' by adding facts $neq(c_1, c_2)$ for each pair of distinct constants $c_1, c_2 \in dom(D)$. We define a new set of EGDs Σ'_E, containing two EGDs, η' and η'', for each $\eta \in \Sigma_E$, built as follows. For each EGD $\eta : \boldsymbol{\phi}(\mathbf{x}) \rightarrow x_i = x_j$ of Σ_E, we let: $\eta' : \boldsymbol{\phi}(\mathbf{x}), null(x_i) \rightarrow x_i = x_j$, and $\eta'' : \boldsymbol{\phi}(\mathbf{x}), null(x_j) \rightarrow x_i = x_j$. The artificial *null* predicate forces x_i (resp. x_j) to bind only to labelled nulls. It is easy to check that assuming $D \cup \Sigma$ is satisfiable, $D \cup \Sigma_T$ is logically equivalent to $D \cup \{\Sigma_T \cup \Sigma'_E\}$, as the only failing cases arise when both x_i and x_j are bound to constants in some EGD and $x_i \neq x_j$ (i.e., hard violations). On the other hand, such EGD activations produce no effects if $x_i = x_j$.

We now construct the set of queries $\mathbf{Q}_V$ that check for hard violations. For each $\eta \in \Sigma_E$, we add to $\mathbf{Q}_V$ the following BCQ: $q \leftarrow \boldsymbol{\phi}(\mathbf{x}), neq(x_i, x_j)$. We have that $D' \cup \{\Sigma_T \cup \Sigma'_E\}$ is always satisfiable, since there cannot be hard violations by construction. Moreover, as Σ_E is harmless, Σ'_E is harmless as well. Thus, by Lemma 3.8, $D' \cup \{\Sigma_T \cup \Sigma'_E\} \models Q_V$ can be checked in PTIME, for each $Q_V \in \mathbf{Q}_V$. □

12 RELAXED WARDED SEMANTICS

Definition 3.10. *Let Σ be a set of warded TGDs, D a database, and T a fact of the chase graph $\mathcal{G}(\Sigma, D)$. Two facts $\boldsymbol{a}$ and $\boldsymbol{b}$ are T-*isomorphic*, if they are isomorphic and have the same track $T = \boldsymbol{track}(\boldsymbol{a}) = \boldsymbol{track}(\boldsymbol{b})$.*

Definition 3.11. *Given a database instance D, a set of warded rules Σ, let $\mathbf{Q}$ be the quotient set $chase(D, \Sigma)/\mathcal{T}$, induced by the T-isomorphism $\mathcal{T}$. We define the relaxed warded semantics $chase^{Bw}(D, \Sigma)$ as the set of all the class representatives of $\mathbf{Q}$, one for each equivalence class of $\mathbf{Q}$.*

13 PROOF OF THEOREM 3.12

Theorem 3.12.a. *Let Σ be a set of warded TGDs, D be a database. If two facts $\boldsymbol{a}$ and $\boldsymbol{b}$ of $\mathcal{G}(\Sigma, D)$ are T-isomorphic, then $subgraph(\mathcal{W}(\mathcal{G}), \boldsymbol{a})$ and $subgraph(\mathcal{W}(\mathcal{G}, \boldsymbol{b}))$ are isomorphic, according to the standard notion of graph isomorphism.*

Proof. It can be easily proven as a special case of Theorem 2 in [4] which holds in the more general case of $\mathbf{a}$ and $\mathbf{b}$ being isomorphic, given that T-isomorphism necessitates isomorphism. □

Theorem 3.12. *Let $\mathbf{S}$ be a database schema and w the maximal arity of its predicates. Given a database D and a set of warded TGDs Σ, both defined for $\mathbf{S}$, let P be the set of pairs $\langle T, \boldsymbol{a} \rangle$ where $T = \boldsymbol{track}(\boldsymbol{a})$. There is a constant δ depending on $\mathbf{S}$, $dom(D)$ and w, such that if $|P| > \delta$, then P contains at least two T-isomorphic facts.*

Proof. In each tree of $\mathcal{W}(D, \Sigma)$, facts can be constructed by permuting at most $w + |\text{dom}(D)|$ terms (at most w new labelled nulls and $\text{dom}(D)$ possible constants) over w positions of $|\mathbf{S}|$ facts. For each tree we have at most $|\mathbf{S}|(w + |\text{dom}(D)|)^w$ non T-isomorphic facts. Tracks are $\binom{|\mathbf{S}|(w+\text{dom}(D))}{2}$, therefore $\delta = |\mathbf{S}|(w + \text{dom}(D))^w \binom{|\mathbf{S}|(w+\text{dom}(D))}{2}$. □

14 PROOF OF THEOREM 3.13

Theorem 3.13. *Given a set $\Sigma = \Sigma_T \cup \Sigma_E$ of warded TGDs and harmless EGDs with respect to Σ, and a database D, if $D \cup \Sigma$ is satisfiable, for every BCQ Q, it holds $chase(D, \Sigma) \models Q$ iff $chase(chase^{BW}(D, \Sigma_T), \Sigma_E) \models Q$.*

Proof. It is sufficient to prove that (a) $chase^{B}(D, \Sigma_T) \models Q$ iff $chase^{BW}(D, \Sigma_T) \models Q$ and then apply the same proof as Theorem 3.7, where $chase^{BW}(D, \Sigma)$ is used in the place of $chase^{B}(D, \Sigma)$. We prove the two directions of the implication. ($\Rightarrow$) We need to show that for every homomorphism h from Q to $chase^{B}(D, \Sigma)$, there is a corresponding homomorphism from Q to $chase^{BW}(D, \Sigma)$. Consider a conjunction of facts $\boldsymbol{\varphi}(\mathbf{x})$ in $chase^{B}(D, \Sigma)$. If $\boldsymbol{\varphi}(\mathbf{x})$ does not contain any pair of T-isomorphic facts, then $\boldsymbol{\varphi}(\mathbf{x}) \in chase^{BW}(D, \Sigma)$. The only case in which $\boldsymbol{\varphi}(\mathbf{x}) \notin chase^{BW}(D, \Sigma)$ is when $\boldsymbol{\varphi}(\mathbf{x})$ contains two T-isomorphic facts φ_i, φ_j and so only one of them, e.g., φ_i is in $chase^{BW}(D, \Sigma)$. In this case, we argue that for every homomorphism from Q to $\boldsymbol{\varphi}(\mathbf{x})$, there is a homomorphism from Q to $\boldsymbol{\varphi'}(\mathbf{x}) = \boldsymbol{\varphi}(\mathbf{x})\{\varphi_j/\varphi_i\}$ ($\boldsymbol{\varphi'}(\mathbf{x})$ is obtained from $\boldsymbol{\varphi}(\mathbf{x})$ by replacing φ_j with φ_i). Let us proceed by contradiction and assume that there does not exist a homomorphism from Q to $\boldsymbol{\varphi'}(\mathbf{x})$. Since φ_i and φ_j are T-isomorphic, they differ at most by some labelled nulls $z_i \neq z_j$, in corresponding positions. In order for z_i and z_j to prevent the existence of homomorphisms from Q to $\boldsymbol{\varphi'}(\mathbf{x})$, z_i and z_j must appear also in other facts of $\boldsymbol{\varphi}(\mathbf{x})$ and $\boldsymbol{\varphi'}(\mathbf{x})$. Let us suppose there is a fact φ_k of $\boldsymbol{\varphi}(\mathbf{x})$ where z_i appears in position $\varphi_k[i]$ and assume there is a corresponding fact φ'_k of $\boldsymbol{\varphi'}(\mathbf{x})$ s.t. z_j is not in position $\phi'_k[i]$. Since φ_i and φ_k, φ_j and φ'_k share a labelled null, respectively, by wardedness it must be that either φ_k resp. φ'_k is a predecessor of φ_i resp. φ_j in the chase, or it is a successor. In the former case, as φ_i and φ_j are T-isomorphic ($\mathbf{track}(\varphi_i) = \mathbf{track}(\varphi_j)$), by Theorem 3.12.a, φ_k and φ'_k must be T-isomorphic, which contradicts our hypothesis that z_i appears in position $\phi_k[i]$ and z_j is not in position $\phi'_k[i]$. In the latter case, from T-isomorphism of φ_i and φ_j, we have that $subgraph(\mathcal{G}, \varphi_i)$ is isomorphic to $subgraph(\mathcal{G}, \varphi'_j)$, and thus φ_k and φ'_k are T-isomorphic ($\mathcal{G}$ is the common chase graph), which also contradicts our hypothesis. Thus we conclude that if there exists a homomorphism from Q to $\boldsymbol{\varphi}(\mathbf{x})$ then there exists a homomorphism from Q to $\boldsymbol{\varphi'}(\mathbf{x})$. ($\Leftarrow$) It directly descends from the hypothesis that $chase^{BW}(D, \Sigma) \subseteq chase^{B}(D, \Sigma)$ and the chase monotonicity. □

15 PROOF OF THEOREM 3.14

Theorem 3.14. *Given a set $\Sigma = \Sigma_T \cup \Sigma_E$ of warded TGDs and harmless EGDs with respect to Σ, and a database D, if $D \cup \Sigma$ is satisfiable, then $chase^{H}(D, \Sigma)$ can be constructed in PTIME in data complexity.*

Proof. We provide a combinatorial proof composed by the following steps:

(1) We first need to show that the overall number of chase steps to construct $chase^{BW}(D, \Sigma)$ is in PTIME data complexity. As $chase^{BW}(D, \Sigma)$ is constructed by the application of TGD steps only, such number is bounded by the size of the chase instance (a single TGD chase step adds a new fact to the instance). For this reason we can simply focus on determining an upper bound for $|chase^{BW}(D, \Sigma)|$.
(2) After determining the upper bound for the number of TGD chase steps to construct $chase^{BW}(D, \Sigma)$, we derive the maximum number of homomorphism checks to consider at each TGD chase step to prove that the procedure for a single TGD chase step is in PTIME data complexity.
(3) We argue that, for each generated fact in the chase procedure, the T-isomorphism condition can be checked in PTIME data complexity. The combination of the steps 1, 2 and 3 is sufficient to prove that $chase^{BW}(D, \Sigma)$ can be constructed in PTIME data complexity.

(4) As a final step, we show that applying the set of harmless EGDs Σ_E to fixpoint over $chase^{Bw}(D,\Sigma)$ to construct $chase^{H}(D,\Sigma)$ is in PTIME data complexity.

First Step. Consider a database D over a schema $\mathbf{S}$, let $\mathbf{w}$ be maximal arity of a predicate in $\mathbf{S}$, $\text{dom}(D)$ be the set of constants in D and δ be a constant such that $\delta{=}|\mathbf{S}|(w+|\text{dom}(D)|)^{w}\binom{|\mathbf{S}|(w+|\text{dom}(D)|)}{2}$. We prove that the size of $chase^{Bw}(D,\Sigma)$ is in PTIME data complexity. By Theorem 3.12, every set of fact pairs $\langle T,\mathbf{a}\rangle$, where $\mathbf{a}$ is a fact and $T=\mathbf{track}(\mathbf{a})$, denoted as P, and such that P$>\delta$ (where $\delta{=}|\mathbf{S}|(w+|\text{dom}(D)|)^{w}\binom{|\mathbf{S}|(w+|\text{dom}(D)|)}{2}$), contains at least two T-isomorphic facts. Since by construction $chase^{Bw}(D,\Sigma)|$ does not contain two T-isomorphic copies, then $|chase^{Bw}(D,\Sigma)|<|\mathbf{S}|(w+|\text{dom}(D)|)^{w}\binom{|\mathbf{S}|(w+|\text{dom}(D)|)}{2}$.

Second Step. Now we prove that the upper bound to all the possible homomorphism checks in the chase procedure for a single TGD chase step is in PTIME data complexity. Let β be the maximal number of body atoms occurring in a TGD in Σ_T. Each TGD in Σ_T admits at most $|chase^{Bw}(D,\Sigma)|^{\beta}$ distinct firing homomorphisms (all the possible disposals of the facts in the chase over β atoms). Since we have shown that $|chase^{Bw}(D,\Sigma)|<|\mathbf{S}|(w+|\text{dom}(D)|)^{w}\binom{|\mathbf{S}|(w+|\text{dom}(D)|)}{2}$ we derive that all the possible homomorphisms are at most $k{=}|\Sigma_T|(|\mathbf{S}|)^{\beta}(w+|\text{dom}(D)|)^{\beta\cdot w}\binom{|\mathbf{S}|(w+|\text{dom}(D)|)}{2}^{\beta}$. This implies that the overall number of homomorphism checks for all the TGD chase steps to construct $chase^{Bw}(D,\Sigma)$ are at most $|chase^{Bw}(D,\Sigma)|\ |\Sigma_T|(|\mathbf{S}|)^{\beta}(w+|\text{dom}(D)|)^{\beta\cdot w}\binom{|\mathbf{S}|(w+|\text{dom}(D)|)}{2}^{\beta}$ which, by replacing the upper bound for $chase^{Bw}(D,\Sigma)$, we derive $|\Sigma_T|(|\mathbf{S}|)^{\beta+1}(w+|\text{dom}(D)|)^{(\beta+1)\cdot w}\binom{|\mathbf{S}|(w+|\text{dom}(D)|)}{2}^{\beta+1}$.

Third Step. Now we prove that the T-isomorphism condition can be checked in PTIME data complexity. As we already discussed, such condition can be checked against facts occurring in a single connected component of the warded forest which, by the proof of Theorem 3.12 are at most $(w+|\text{dom}(D)|)^{w}$ and bounded by k. This implies that the overall number of checks for the T-isomorphism condition is at most $|\mathbf{S}|(w+|\text{dom}(D)|)^{2w}\binom{|\mathbf{S}|(w+|\text{dom}(D)|)}{2}$.

Fourth Step. As a final step we prove that applying the set of harmless EGDs Σ_E to fixpoint from $chase^{Bw}(D,\Sigma)$ is in PTIME data complexity. Let us consider the set of harmless EGDs Σ_E and let γ be the maximum number of body atoms of an EGD in Σ_E. Equivalently to warded TGDs, we can assume that the number of homomorphism checks to perform for all the EGD chase steps are at most $|chase^{Bw}(D,\Sigma)|\ |\Sigma_E|(|\mathbf{S}|)^{\gamma}(w+\text{dom}(D))^{\gamma\cdot w}\binom{|\mathbf{S}|(w+\text{dom}(D))}{2}^{\gamma}$ which again, is bounded by $|\Sigma_E|(|\mathbf{S}|)^{\gamma+1}(w+|\text{dom}(D)|)^{(\gamma+1)\cdot w}\binom{|\mathbf{S}|(w+\text{dom}(D))}{2}^{\gamma+1}$. Moreover, the application of a single EGD chase step requires the substitution of all the occurrences of a labelled null in the chase instance. For such procedure, at most $(w+|\text{dom}(D)|)^{w}$ facts need to be checked, as by wardedness, a labelled null is propagated within a single connected component of the warded forest (and this is the only set of facts that needs to be checked). By combining the complexity of steps 1-3 and step 4 we conclude that $chase^{H}(D,\Sigma)$ can be constructed in PTIME data complexity. □

16 PROOF OF THEOREM 3.15

Theorem 3.15. *If a set of TGDs and EGDs $\Sigma=\Sigma_T\cup\Sigma_E$ is separable then Σ_E is harmless wrt Σ (and not vice versa).*

Proof. We first show that EGD separability implies harmlessness. By separability, for every BCQ Q, it holds $chase(D,\Sigma)\models Q$ iff $chase(D,\Sigma_T)\models Q$. It is sufficient to choose the identity homomorphism for h to derive Definition 3.2 and conclude Σ_E is harmless wrt Σ. Example 16.1 is sufficient to show the reverse does not hold as, in such case, Σ is not separable, while Σ_E is harmless wrt Σ. □

Example 16.1.

$$component(x) \rightarrow \exists z\ component(z), partOf(x, z) \tag{σ_1}$$

$$partOf(x, v), partOf(x, w) \rightarrow v = w \tag{η_1}$$

$$component(x), component(y),$$
$$tag(x, y), partOf(x, v), partOf(y, w) \rightarrow v = w \tag{η_2}$$

Components are parts of other components (σ_1). Every component is part of one single component (η_1) and two components sharing a tag are part of the same component (η_2). ■

17 FURTHER REAL-WORLD USE CASES OF HARMLESS EGDS

This section expands Section 5.4 by providing further real-world problems that can be effectively modeled with a reasoning approach based on an interaction between warded TGDs and harmless EGDs.

17.1 Strongly Connected Components (SCC)

As a further example for graph problems that can be effectively modeled with an interaction between warded TGDs and harmless EGDs we consider the strongly connected graph problem. Specifically, a directed graph is strongly connected if every node is reachable from another (and vice-versa). A strongly connected component (SCC) of a directed graph is a set of mutually reachable nodes. The aim of our reasoning settings is to determine whether an input directed graph is strongly connected and to identify the nodes in the same strongly connected component (SCC).

Example 17.1. The strongly connected component problem modeled by the following set of TGDs and harmless EGDs.

$$Edge(x, y) \rightarrow \exists c_1, c_2\ SCC(x, c_1), SCC(y, c_2) \tag{σ_1}$$

$$Edge(x, y) \rightarrow Path(x, y) \tag{σ_2}$$

$$Path(x, y), Edge(y, z) \rightarrow Path(x, z) \tag{σ_3}$$

$$Path(x, y), Path(y, x), SCC(x, z_1), SCC(y, z_2) \rightarrow z_1 = z_2 \tag{η}$$

Each node is assigned to a strongly connected component (σ_1). For each edge connecting the node x to y there exists a path from x to y (σ_2). If there is a path connecting x to y and an edge connecting y to z, there exists a path from x to z (σ_3). Strongly connected components for the nodes x and y are merged whether there is a path from x to y and from y to x as transitively, it is possible to reach every node in the SCC of x from y and viceversa (η).

$$q \leftarrow SCC(x, z_1), SCC(y, z_2), z_1 \neq z_2 \tag{Q_1}$$

$$q(x, y) \leftarrow SCC(x, z), SCC(y, z) \tag{Q_2}$$

By the BCQ Q_1 we check whether the graph is not strongly connected. If the answer is negative the CQ Q_2 extracts the pair of nodes in the same strongly connected component (SCC).

17.2 Data Matching

To further complement our use case exploration of harmless EGDs, in this section, we provide a real-world application of harmless EGDs in the context of data matching (record linkage) problems. Data matching is the task of finding

records that refer to the same entity across different data sources. It can be seen as a specific step in a data integration process. In this section we provide a reasoning-based solution to a data matching problem in the Airbnb domain, a famous platform for short-term house rentals. Specifically, we propose a deterministic blocking phase (i.e., the process of grouping objects which are likely to be the same real-world entity) which aims to identify possible duplicated listings for apartment rentals in the Airbnb website. For our reasoning settings we consider the following relational schema containing information about house hosts and apartment listings:

- *Listing*(URL, name, description, address, latitude, longitude, n_beds, n_bathrooms, price, owner, neighbor)
- *Host*(host_id, host_name, ...)
- *Kindred*(host_id1, host_id2, ...)

Example 17.2. A data matching scenario in the domain of Airbnb modeled by a set of warded TGDs and harmless EGDs. The goal of the reasoning scenario is to identify the listings on the Airbnb website that refer to the same apartment.

$$Host(host\_id, \_) \rightarrow \exists f\ Family(host\_id, f) \qquad (\sigma_1)$$

$$Listing(URL, name, \dots) \rightarrow \exists id\ Listing\_X(id, URL, name, \dots) \qquad (\sigma_2)$$

$$Listing\_X(listing\_id, \dots, owner, \dots), Family(owner, f) \rightarrow Own(owner, listing\_id, f) \qquad (\sigma_3)$$

$$Family(host\_id_1, f_1), Kindred(host\_id_1, host\_id_2, \dots), Family(host\_id_2, f_2) \rightarrow f_1 = f_2 \qquad (\eta_1)$$

$$Listing\_X(id_1, \dots, name, \dots, owner), Listing\_X(id_2, \dots, name, \dots, owner) \rightarrow id_1 = id_2 \qquad (\eta_2)$$

$$Listing\_X(id_1, \dots, lat, long, owner, \dots), Listing\_X(id_1, \dots, lat, long, owner, \dots) \rightarrow id_1 = id_2 \qquad (\eta_3)$$

For each Airbnb host there exists a family f (σ_1). Each Airbnb listing has a distinct id (σ_2). A host in family is the owner of a specific apartment exposed in the Airbnb website by a listing (σ_3). If two hosts are kindred they belong to the same family f (η_1). If two listings have the same name and owner they refer to the same apartment and their assigned ids are merged (η_2). If two listings have the same latitude, longitude and owner it is likely that they refer to the same apartment (η_3).

$$q(name_1, desc_1, name_2, desc_2, \dots) \leftarrow Listing\_X(id, name_1, desc_1, \dots), Listing\_X(id, name_2, desc_2, \dots) \qquad (Q_1)$$

$$q(listing_1, listing_2) \leftarrow Own(owner_1, listing_1, F), Own(owner_2, listing_2, F) \qquad (Q_2)$$

By the CQ Q_1 we extract the listings referring to the same apartment. Conversely, the CQ Q_2 identifies the listings for apartments owned by the same family.

17.3 Data Fusion

In this section we present a common data fusion scenario in a Geographic Information System (GIS). In such systems, data are always acquired by several data sources located in distinct area and require to be combined together into a single database. In our settings, the GPS sensors of the devices communicate periodically to three distinct satellites, which on turn, send data to the central application server of the GIS. We also assume that multiple cameras are located in distinct town blocks and periodically send geotagged pictures to the central application server. The task of our system is to reconstruct the paths of the devices transiting across the town blocks through geotagged pictures. We propose a reasoning-based solution which exploits an interaction between harmless EGDs and warded TGDs. As already

mentioned, all the acquired data by the sensors and cameras are stored into a single database of our application server with the following schema:

- *GPS_Satellite_A*(timestamp, sensorId, distance)
- *GPS_Satellite_B*(timestamp, sensorId, distance)
- *GPS_Satellite_C*(timestamp, sensorId, distance)
- *Picture_A*(timestamp, cameraId, content, ...)
- *Picture_B*(timestamp, cameraId, content, ...)
- *Picture_C*(timestamp, cameraId, content, ...)
- *Cameras*(timestamp, cameraId, ...)
- *Connected*(cameraId1, cameraId2)

In our reasoning settings, the GPS data are combined together in the central application server and the positions of the devices are extracted by adopting the GPS triangulation approach. On the other hand, the pictures taken by the cameras are integrated in a single data source and associated to the devices transiting in the captured areas. This is possible by matching the picture coordinates and the transit timestamps. Finally, by computing the answers to specific CQs, the system is able to reconstruct the paths of the devices transiting across the town blocks with geotagged pictures.

Example 17.3. A data fusion scenario in the IoT domain modeled with a set of warded TGDs and harmless EGDs.

$$GPS\_Satellite\_A(timestamp, deviceId, dist_1), GPS\_Satellite\_B(timestamp, deviceId, dist_2),$$
$$GPS\_Satellite\_C(timestamp, deviceId, dist_3) \rightarrow Positions(deviceId, timestamp, dist_1, dist_2, dist_3) \quad (\sigma_1)$$

$$Positions(deviceId, timestamp, dist_1, dist_2, dist_3), coord = compute\_coord(dist_1, dist_2, dist_3)$$
$$\rightarrow Coordinates(deviceId, timestamp, coord) \quad (\sigma_2)$$

GPS records stored in different tables (i.e., acquired by distinct satellites) are merged into a single data source with the aim of inferring the location of a device at a certain time (σ_1). The position of a device is computed with the GPS triangulation approach adopting the three distances to the satellites (σ_2).

$$PicturesA(timestamp, cameraId, content, \dots), Cameras(cameraId, coordinates, \dots)$$
$$\rightarrow \exists X\ Pictures(X, timestamp, cameraId, content, \dots, coordinates) \quad (\sigma_3)$$

$$PicturesB(timestamp, cameraId, content, \dots), Cameras(cameraId, coordinates, \dots)$$
$$\rightarrow \exists X\ Pictures(X, timestamp, cameraId, content, \dots, coordinates) \quad (\sigma_4)$$

$$PicturesC(timestamp, cameraId, content, \dots), Cameras(cameraId, coordinates, \dots)$$
$$\rightarrow \exists X\ Pictures(X, timestamp, cameraId, content, \dots, coordinates) \quad (\sigma_5)$$

Pictures data of different sources (A, B and C) are matched with to camera locations thanks to the coordinates field and then integrated into a single table (σ_3, σ_4, σ_5).

$$Pictures(X, timestamp, cameraId, content, \dots, coord), Coordinates(deviceId, timestamp, coord)$$
$$\rightarrow \exists Y\ PSnapshot(deviceId, cameraId, coord, timestamp, Y, content, \dots) \quad (\sigma_6)$$

A picture is assigned to a specific device if they share the same coordinates. Observe that a picture can be assigned to multiple devices at a certain time (σ_6).

$$Pictures(X_1, timestamp, cameraId_1, \ldots, coordinates), Pictures(X_2, timestamp, cameraId_2, \ldots, coordinates),$$
$$cameraId_1 \neq cameraId_2 \rightarrow X_1 = X_2 \quad (\eta_1)$$
$$PSnapshot(deviceId, cameraId_1, coord, tstamp, Y_1, \ldots), PSnapshot(deviceId, cameraId_2, coord, tstamp, Y_2, \ldots),$$
$$cameraId_1 \neq cameraId_2 \rightarrow Y_1 = Y_2 \quad (\eta_2)$$
$$PSnapshot(deviceId, camId_1, coord_1, tstamp_1, Y_1, \ldots), PSnapshot(deviceId, camId_2, coord_2, tstamp_2, Y_2, \ldots),$$
$$Connected(camId_1, camId_2), camId_1 \neq camId_2, coord_1 \neq coord_2, tstamp_2 > tstamp_1 \rightarrow Y_1 = Y_2 \quad (\eta_3)$$

Two picture records sharing the same coordinates and the same timestamps are assigned to the same group as they likely capture data referring to the same element (η_1). Two picture records refer to the same device if they are taken in the same time (i.e., timestamp field) by distinct cameras placed in the same location (η_2). Pictures taken by different cameras placed in close town blocks, captured a transiting device. Hence the records are assigned to the same group (η_3).

$$q(deviceId, coord, \ldots) \leftarrow PSnapshot(deviceId, \ldots, coord, Y, \ldots), PSnapshot(deviceId, \ldots, coord, Y, \ldots) \quad (Q_1)$$
$$q(content_1, content_2) \leftarrow Pictures(X, \ldots, content1), Pictures(X, \ldots, content_2) \quad (Q_2)$$
$$q(deviceId, coord_1, coord_2, cont_1, cont_2) \leftarrow$$
$$PSnapshot(deviceId, camId_1, coord_1, tstamp_1, cont_1, Y), PSnapshot(deviceId, camId_2, coord_2, tstamp_2, cont_2, Y),$$
$$Connected(camId_1, camId_2), camId_1 \neq camId_2, coord_1 \neq coord_2, timestamp_2 > timestamp_1 \quad (Q_3)$$

The CQ Q_1 extracts all the geotagged pictures referring to the same device taken in the same location. The CQ Q_2 infers the pictures taken by different cameras in the same location with the same timestamp (as a binary relation with additional fields). Finally, the CQ Q_3 reconstruct the paths of the devices transiting across the town blocks with geotagged pictures.

REFERENCES

[1] Marcelo Arenas, Georg Gottlob, and Andreas Pieris. Expressive languages for querying the semantic web. In *PODS*, pages 14–26, 2014.

[2] Marcelo Arenas, Georg Gottlob, and Andreas Pieris. Expressive languages for querying the semantic web. *ACM Trans. Database Syst.*, 43(3):13:1–13:45, 2018.

[3] Teodoro Baldazzi, Luigi Bellomarini, Emanuel Sallinger, and Paolo Atzeni. Eliminating harmful joins in warded datalog+/-. In *RuleML+RR*, Lecture Notes in Computer Science. Springer, 2021.

[4] Luigi Bellomarini, Emanuel Sallinger, and Georg Gottlob. The vadalog system: Datalog-based reasoning for knowledge graphs. *PVLDB*, 11(9):975–987, 2018.

[5] Andrea Calì, Georg Gottlob, Giorgio Orsi, and Andreas Pieris. On the interaction of existential rules and equality constraints in ontology querying. In *Correct Reasoning*, volume 7265 of *Lecture Notes in Computer Science*, pages 117–133. Springer, 2012.

[6] Beeri Catriel and Moshe Y. Vardi. The implication problem for data dependencies. *Automata, Languages and Programming*, pages 73–85, 1981.

[7] Evgeny Dantsin, Thomas Eiter, Georg Gottlob, and Andrei Voronkov. Complexity and expressive power of logic programming. *ACM Comput. Surv.*, 33(3):374–425, 2001.

[8] Georg Gottlob and Andreas Pieris. Beyond sparql under owl 2 ql entailment regime: Rules to the rescue. In *Proceedings of the 24th International Conference on Artificial Intelligence*, IJCAI'15, page 2999–3007. AAAI Press, 2015.

[9] Georg Gottlob and Andreas Pieris. Beyond SPARQL under OWL 2 QL entailment regime: Rules to the rescue. In *IJCAI*, pages 2999–3007, 2015.