Approximate Functional Dependencies—Implication Problem Revisited

Nicolas Fröhlich[1,*,†], Matilda Häggblom[2,†], Åsa Hirvonen[2,†] and Minna Hirvonen[1,†]

[1]*Leibniz Universität Hannover, Germany*
[2]*University of Helsinki, Finland*

Abstract

Functional dependencies are an important and well-studied class of database constraints that correspond to a notion expressed by dependence atoms in team logic. In practice, data often contain errors, so in some cases it might be useful to allow the database to have a small number of tuples that violate the desired dependency. Väänänen (2017) studied the axiomatization of a notion of approximate dependence that specifies for each dependence atom how much of the database can be disregarded. We demonstrate that the interaction of approximate dependence atoms is more complicated than previously thought in the sense that there is a semantic consequence that is not captured by the inference rules introduced before. We show that Väänänen's axiomatisation is still complete in the restricted case of unary dependencies. We also consider the complexity of model checking for approximate dependence: it is NP-complete for disjunctions of two atoms and LOGSPACE-hard for individual atoms.

Keywords

Database, functional dependency, approximation, team semantics, axiomatisation, model checking.

1. Introduction

Functional dependencies were introduced in database theory by Codd in the 1970's [1], and axiomatised by Armstrong [2] a few years later. In the unirelational case they are equivalent to dependence atoms in team logic. Teams were originally defined by Hodges [3, 4] to give a compositional semantics for so-called independence-friendly logic. Väänänen [5] introduced dependence logic, allowing the explicit study of relationships between variables.

The rise of big data offers strong motivation to study dependencies that disregard statistical outliers in the data set. Approximate notions of dependence are also motivated by considerations of natural language, such as 'most', 'all but a few', and were formalised for functional dependencies by Kivinen and Mannila [6] with various measures of the part of the information allowed to be erroneous. Väänänen adapted one of the definitions as approximate dependence atoms in the team semantic setting [7], and proposed an axiomatisation of its implication problem.

Unfortunately, the axiomatisation in [7] disregards a subtle point in dependence that only turns up in the approximate notion. Where in pure functional dependencies, a tuple yz depends on x exactly when both y and z depend on x, in an approximate version, where small errors are allowed, the distribution of errors between variables introduces interplay between the variables that is hard to axiomatise. In this paper, we illustrate this phenomenon by presenting a 'missing rule' - a semantic consequence that cannot be obtained from the axiomatisation in [7].

Väänänen's axiomatisation is still complete in the restricted case of *unary dependencies*, where simple dependence statements 'y depends on x' are considered for single variables x and y. We extract the necessary rules for unary and constant dependencies, and prove completeness (there was an error in

LINDA'26, July 24, 2026, Lisbon, Portugal
*Corresponding author.
†These authors contributed equally.
nicolas.froehlich@thi.uni-hannover.de (N. Fröhlich); matilda.haggblom@helsinki.fi (M. Häggblom); asa.hirvonen@helsinki.fi (Å. Hirvonen); minna.hirvonen@thi.uni-hannover.de (M. Hirvonen)
0009-0003-5413-1823 (N. Fröhlich); 0009-0003-6289-7853 (M. Häggblom); 0000-0003-2149-4153 (Å. Hirvonen); 0000-0002-2701-9620 (M. Hirvonen)

the proof in [7]). Such axiomatizations could help speed up algorithms for discovering approximate dependencies in databases (see, e.g., [8]), with applications in data mining.

We also examine the complexity of the model checking problem for approximate dependence.

2. Definitions

We recall basic team semantic definitions together with the approximate dependence atoms from [7].

Let D and M be sets of variables and values, respectively. An assignment s over D is a function $s\colon D \to M$. A team X over D is a set of assignments $s\colon D \to M$. For a team X over D and a tuple $x = x_1 \dots x_n$ of variables from D, we define the notation $X[x] := \{s(x) \in M^n \mid s \in X\}$, where $s(x) = s(x_1) \dots s(x_n)$. For $x = x_1 \dots x_n$ and $y = y_1 \dots y_m$, we write xy for the tuple $x_1 \dots x_n y_1 \dots y_m$. The notation $|x|$ refers to the length of the tuple x, e.g, $|x| = n$ for $x = x_1 \dots x_n$.

Let x and y be finite tuples of variables from D such that y is nonempty, i.e., $|y| \geq 1$. An expression $=(x, y)$ is called a *dependence atom*. The satisfaction relation between a team X over D and the atom $=(x, y)$ is defined by

$$X \models =(x, y) \text{ if and only if for all } s, s' \in X,\ s(x) = s'(x) \text{ implies } s(y) = s'(y).$$

Since $X \models =(x, y)$ if and only if there is a function $f\colon X[x] \to X[y]$ such that $f(s(x)) = s(y)$ for all $s \in X$, dependence atoms are often called *functional dependencies*, especially in database theory. Note that the tuple x is allowed to be empty[1], and in that case, we write $=(y)$ instead of $=(, y)$. The atom $=(y)$ is called the *constancy atom*, as it states that the value of y is constant in the team. Let $\Sigma \cup \{\tau\}$ be a set of atoms with variables from D. We write $X \models \Sigma$, if $X \models \sigma$ for all $\sigma \in \Sigma$. We say that Σ *semantically entails* τ, written as $\Sigma \models \tau$, if for all X over D, we have that $X \models \Sigma$ implies $X \models \tau$.

We consider a notion of *approximate dependence* that states that a dependence must hold in a team when we are allowed to remove some assignments (usually a small portion of the whole team).

Definition 2.1 ([7]). *Let p be a real number, $0 \leq p \leq 1$. For finite teams, $X \models =_p(x, y)$ if there is a subteam $Y \subseteq X$, $|Y| \leq p \cdot |X|$, such that $X \setminus Y \models =(x, y)$.*

The following set of rules is sound for approximate dependencies.

Definition 2.2 ([7]). *Rules $A1$-$A7$ form the system $\mathbf{A}$ for approximate dependencies.*

- *(A1)* $=_0(xy, x)$.
- *(A2)* $=_1(x, y)$.
- *(A3) If* $=_p(x, yv)$, *then* $=_p(xu, y)$.
- *(A4) If* $=_p(x, y)$, *then* $=_p(xu, yu)$.
- *(A5) If* $=_p(xu, yv)$, *then* $=_p(ux, yv)$ *and* $=_p(xu, vy)$.
- *(A6) For* $r = min\{p + q, 1\}$, *if* $=_p(x, y)$ *and* $=_q(y, v)$, *then* $=_r(x, v)$.
- *(A7) For* $p \leq q \leq 1$, *if* $=_p(x, y)$, *then* $=_q(x, y)$.

For a set of approximate dependencies Σ and a system of rules $\mathbf{B}$, we write $\Sigma \vdash_{\mathbf{B}} =_p(x, y)$ if we can apply the rules in $\mathbf{B}$ to the dependencies in Σ to derive the conclusion $=_p(x, y)$. A system $\mathbf{B}$ is sound if whenever $\Sigma \vdash_{\mathbf{B}} =_p(x, y)$, also $\Sigma \models =_p(x, y)$. As shown in [7], the rules are sound, but we challenge the claim that they are *complete* in the sense claimed in [7], i.e., that for a finite set of approximate dependence atoms, $\Sigma \models =_p(x, y)$ would imply that $\Sigma \vdash_{\mathbf{A}} =_p(x, y)$.

First, let us note a minor issue that arises when the dependence is defined on sequences rather than on sets of variables/attributes, as is common in the database theory literature. Using system $\mathbf{A}$, we cannot derive $=_p(x, xy)$ from $=_p(x, y)$. Instead, we can derive $=_p(x, y) \vdash_{\mathbf{A}} =_p(xx, xy)$, so if we consider xx as different from x, the set of rules is not complete even for usual dependencies with approximation 0. To solve this, as pointed out in [9], we can replace the rules $A3$ and $A4$ with the rule A_{34}:

$$(A_{34}) \quad \text{If } =_p(x, y), \text{ then } =_p(x, xy).$$

[1] In the dependence atom $=(x, y)$, the tuple y is assumed to be nonempty, because otherwise the atom $=(x, y)$ would be trivially satisfied by any team over D that contains the variables x.

Table 1
Example team for Example 3.1. The team satisfies all approximation atoms in Σ, while also satisfying $=_{\frac{5}{16}}(x,y)$. However $=_{\frac{5}{16}}(x,y)$ cannot be derived from Σ given the rules in system $\mathbf{A}^*$.

	x	v_1	v_2	v_3	y	z_0
s_0	0	0	0	1	1	0
s_1	0	0	0	2	2	1
s_2	0	0	0	3	3	2
s_3	0	1	0	4	4	3
s_4	0	0	1	0	5	4
s_5	0	0	0	0	0	5
s_6	0	0	0	0	0	6
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
s_{15}	0	0	0	0	0	15

System $\mathbf{A}^*$ consisting of rules $A1, A2, A_{34}, A5, A6$ and $A7$ is complete for usual dependencies sensitive to repetition of variables (see e.g., [10] for a completeness proof in the team semantic setting). However, we show in the next section that there are semantic entailments specific to the approximate setting that are not derivable even in this modified set of rules $\mathbf{A}^*$.

3. Missing Rule

We aim to show that there is a semantic entailment among approximate dependence atoms that is not derivable using rules from the system $\mathbf{A}^*$, proving that the system is incomplete. First, we exemplify this semantic entailment by examining a specific team.

Example 3.1. *Let $\Sigma = \{=_0(x), =_{\frac{1}{8}}(x, v_1v_2), =_{\frac{2}{8}}(x, v_1v_3), =_{\frac{2}{8}}(v_1, y), =_0(v_2v_3, y)\}$ be a set of approximate atoms. Consider the team X depicted in Table 1. We have that $X \models \Sigma$, while having to remove the maximum number of assignments in each approximation atom in Σ. Next, consider the dependence atom $=(x,y)$. For $=(x,y)$ to be true, five assignments in X need to be removed; therefore $X \models =_{\frac{5}{16}}(x,y)$ holds. However $=_{\frac{5}{16}}(x,y)$ cannot be derived by the rules in system $\mathbf{A}^*$. The best approximation atom derivable is $=_{\frac{6}{16}}(x,y)$ via transitivity of $=_{\frac{1}{8}}(x, v_1v_2)$ and $=_{\frac{2}{8}}(v_1, y)$. While $X \models =_{\frac{6}{16}}(x,y)$ is obvious, the ratio $\frac{6}{16}$ is clearly not optimal.*

The next result shows that this example is not an isolated case, but that all teams satisfying Σ must also satisfy $=_{\frac{5}{16}}(x,y)$.

Proposition 3.2. *Let $\Sigma = \{=_0(x), =_{\frac{1}{8}}(x, v_1v_2), =_{\frac{2}{8}}(x, v_1v_3), =_{\frac{2}{8}}(v_1, y), =_0(v_2v_3, y)\}$ be a set of approximate atoms. If $X \models \Sigma$ for any team X, then $X \models =_{\frac{5}{16}}(x,y)$.*

However, since $\Sigma \not\vdash_{\mathbf{A}^*} =_{\frac{5}{16}}(x,y)$, we conclude that $\mathbf{A}^*$ is an incomplete set of rules.

It is worth noting that the missing rule does not only show incompleteness in the case with tuples as *dependent* variables (i.e., in the right part of the atom). If one studies *right unary* dependence atoms, i.e., atoms of the form $=_p(x_1 \dots x_n, y_1)$, one can simulate atoms of the form $=_p(x, y_1y_2)$ by introducing a new variable z and using dependencies $=_0(z, y_1)$, $=_0(z, y_2)$, $=_0(y_1y_2, z)$, $=_p(x, z)$.

4. Restricted Completeness Theorem

We prove the completeness of the original system in Definition 2.2 for a restricted set of approximate dependence atoms. Namely, when we restrict to rational approximations $p \in \mathbb{Q} \cap [0,1]$ and to unary and so-called 1-constancy atoms of the form $=_p(x,y)$, where $|x| \leq 1$ and $|y| = 1$.

Table 2
Suppose that $=_{\frac{1}{4}}(x,y)$ is not derivable from Σ using rules from system $\mathbf{A}^1$, where Σ is $\{=_{\frac{1}{4}}(x,w), =_{\frac{1}{4}}(w,y)\}$. Then $d(x)=0$, $d(w)=\frac{1}{4}$, and $d(y)=\frac{1}{2}$ and we can construct counterexample teams X and Y as illustrated.

X	x	w	y	z_0
s_0	0	0	0	0
s_1	0	1	1	1
s_2	0	1	2	2
s_3	0	1	2	3
s_4	0	1	2	4

Y	x	w	y
s_0	0	0	0
s_1	0	1	1
s_2	0	1	2
s_3	a	a	a
s_4	b	b	b

Due to the arity restrictions, we can omit rules $A4$ and $A5$. We denote the remaining rules restricted to unary and 1-constancy atoms by $A1^1$, $A2^1$, $A3^1$, $A6^1$ and $A7^1$ and obtain system $\mathbf{A}^1$. In particular, $A1^1$ takes the simple form '$=_0(x,x)$', and $A3^1$ is of the form 'if $=_p(y)$, then $=_p(u,y)$'.

Theorem 4.1. *Let $\Sigma \cup \{=_p(x,y)\}$ be a finite set of unary and 1-constancy rational approximate dependence atoms. If $\Sigma \models =_p(x,y)$, then $\Sigma \vdash_{\mathbf{A}^1} =_p(x,y)$.*

To prove completeness under these restrictions, we construct a counterexample team X similar to the one in [7], but with some core differences, one of which we discuss next. Let $\Sigma \cup \{=_p(x,y)\}$ be as in the above Theorem. Our goal is to show, under the assumption $\Sigma \nvdash_{\mathbf{A}^1} =_p(x,y)$, that there is a team X that satisfies all atoms in Σ but not $=_p(x,y)$. Like in [7], for each variable u we find the smallest approximation $d(u)$ for which $=_{d(u)}(x,u)$ is derivable from Σ. Differing from [7], we can find the least common denominator $n-1$ for the approximations appearing in $\Sigma \cup \{=_p(x,y)\}$, since they are rational numbers. We then construct a team of size n such that the approximations $d(u)=\frac{k}{n-1}$ are *uniformly* encoded with a slightly better approximation, namely $\frac{k}{n}$. This type of uniformity is missing from [7], making that completeness proof erroneous even for unary atoms, possibly because of the attempt to cover real-numbered approximations. Two examples of counterexample teams are presented in Table 2, one of which takes advantage of a dummy variable z_0 and the other does not. A similar construction for unary approximate *inclusion atoms* with rational approximations also appeared in [11].

5. Model Checking

Model checking is a central decision problem in every logic. It asks the simple question whether a formula of the logic is true in a structure. One often considers the *data complexity* variant where the formula is fixed and the input consists of only the structure. For dependence logic, model checking is NP-complete in general, even for quantifier-free disjunctions of only three dependence atoms, but in NL, when restricted to disjunctions of two atoms [12]. However, with approximations even simpler formulas can have intractable model checking, as was shown for the approximate operator in [13]. Our first result demonstrates that with a single approximate dependence atom, disjunctions of just two atoms become NP-complete. In the setting of dependence logic, disjunctions are not defined globally over the whole team but rather split the team into two parts, each satisfying one of the two disjuncts. Formally, $X \models \phi_1 \vee \phi_2$ if there are teams X_1, X_2 such that $X = X_1 \cup X_2$ and $X_i \models \phi_i$ for $i \in \{1,2\}$.

Theorem 5.1. *The model checking problem for $=_{\frac{3}{13}}(u,v) \vee =_0(x,y)$ is NP-complete.*

Next, consider the complexity of a single approximate dependence atom. It is well known that single dependence atoms are first-order definable. This is no longer true for approximate dependence atoms.

Theorem 5.2. *The model checking problem for $=_{\frac{1}{2}}(x,y)$ is LOGSPACE-hard.*

6. Conclusion and Future Work

We introduced a semantic entailment that cannot be deduced by using system $\mathbf{A}^*$, meaning that the system is not complete in the general case of approximate dependencies. It seems likely that there are other similar semantic entailments involving different approximations, so the system might actually have several missing rules. This raises the question whether the implication problem is finitely axiomatisable or whether every complete system necessarily involves an infinite number of rules. If the problem is not finitely axiomatisable, a natural goal for future work would be to prove that this is indeed the case. It might still be possible to represent the axioms in a nice way that expresses the infinite rules in the form of a rule schema, e.g., as in [14, 15, 16]. Investigating this is also a relevant question for further research.

Since the implication problem for dependencies with real (or even rational) number approximations seems difficult to axiomatise, one may ask if this has something to do with some specific properties of these structures. Therefore, it might be useful to investigate the approximate dependence in the setting of monoids, where we may consider structures with different properties. A monoid is a tuple $K = (K, +, 0_K)$ where K is a set, $+$ is an associative binary operation on K, and 0_K is an identity element of $+$. A monoid K is *positive* if $a + b = 0_K$ implies $a = 0_K$ and $b = 0_K$, and *commutative* if $a + b = b + a$ for all $a, b \in K$. An *ordered* monoid is a commutative monoid K with a partial order $\leq$ such that $0_K \leq a$ for all $a \in K$, and $a \leq b$ implies $a + c \leq b + c$ for all $a, b, c \in K$. The idea is that we consider a totally ordered positive commutative monoid $K = (K, \leq, +, 0_K)$, and for a finite team $X \subseteq \{s \mid s\colon D \to M\}$, define a K-team $\mathbb{X}$ as a function $\mathbb{X}\colon X \to K$. A K-team is essentially an annotated database relation or a K-relation, as introduced in [17] in the setting of semiring provenance. Denote $\mathrm{supp}(\mathbb{X}) := \{s \in X \mid \mathbb{X}(s) \neq 0_K\}$. Then for any $a \in K$, we can define a K-approximate dependence atom $=_a(x, y)$ with the following satisfaction relation: $\mathbb{X} \models =_a(x, y)$ iff there is $Y \subseteq X$ for which $\sum_{s \in Y} \mathbb{X}(s) \leq a$ and $\mathrm{supp}(\mathbb{X} \upharpoonright_{X \setminus Y}) \models =(x, y)$. Note that we define $\sum_{s \in \emptyset} \mathbb{X}(s) = 0_K$, so $\mathbb{X} \models =_{0_K}(x, y)$ iff $\mathrm{supp}(\mathbb{X}) \models =(x, y)$. If K is the Boolean monoid $\mathbb{B} = (\{0, 1\}, \leq, \vee, 0)$, the K-approximate dependencies correspond to the usual dependencies, where each $=_0(x, y)$ corresponds to $=(x, y)$ and each $=_1(x, y)$ is trivially satisfied. If K is the probability monoid $\mathbb{P} = ([0, 1], \leq, +, 0)$, where $\leq$ is the usual order of the unit interval $[0, 1]$ and $a + b = \max\{a + b, 1\}$, then the K-approximate dependencies correspond to the approximate dependencies over K-teams $\mathbb{X}$ that are uniform distributions, i.e., $\mathbb{X}(s) = 1/|X|$ for all $s \in X$. Different dependency notions over K-teams have been studied, e.g., in [18, 19][2].

In the setting of monoids, the satisfaction of the K-approximate dependencies is defined in such a way that the number of tuples that can be removed is measured as an absolute value instead of a team ratio. One possible line of future research on real approximations is to examine whether considering approximation as an absolute value instead of a ratio could help with obtaining a complete axiomatisation. Already in [6], variants of both absolute value approximations and ratio approximations were introduced. One can also consider other definitions, such as a *local* variant where $X \models =_p(x, y)$ if there is a subteam $Y \subseteq X$, $|Y| \leq p \cdot |X \upharpoonright_{\{xy\}}|$, such that $X \setminus Y \models =(x, y)$, where $X \upharpoonright_{\{xy\}}$ is the team restricted to the variables x and y. With this definition, $\{=_p(x, w), =_q(w, y)\} \not\models =_{p+q}(x, y)$, meaning any complete system, even restricted to unary atoms, would be different from the one in [7].

To take advantage of the team semantic framework, we could situate the approximate dependence atom in a language with connectives and quantifiers as in first-order dependence logic [5]. Some first steps in this direction are taken in [13]. With our gained understanding of unary and 1-constancy approximate dependence atoms, they could serve as the first building blocks of an approximate variant of first-order dependence logic.

[2]Note that these works also consider (conditional) independence atoms whose semantics is defined using both addition and multiplication, so their structure K is a *semiring* instead of a monoid.

References

[1] E. F. Codd, Further normalization of the data base relational model, IBM Research Report, San Jose, California RJ909 (1971).

[2] W. W. Armstrong, Dependency structures of data base relationships, in: IFIP Congress, 1974. URL: https://api.semanticscholar.org/CorpusID:38788061.

[3] W. Hodges, Compositional semantics for a language of imperfect information, Logic Journal of IGPL 5 (1997) 539–563. doi:10.1093/jigpal/5.4.539.

[4] W. Hodges, Some strange quantifiers, in: J. Mycielski, G. Rozenberg, A. Salomaa (Eds.), Structures in Logic and Computer Science: A Selection of Essays in Honor of A. Ehrenfeucht, Lecture Notes in Computer Science, Springer, Berlin, Heidelberg, 1997, pp. 51–65. doi:10.1007/3-540-63246-8_4.

[5] J. Väänänen, Dependence Logic: A New Approach to Independence Friendly Logic, London Mathematical Society Student Texts, Cambridge University Press, 2007.

[6] J. Kivinen, H. Mannila, Approximate inference of functional dependencies from relations, Theoretical Computer Science 149 (1995) 129–149. doi:10.1016/0304-3975(95)00028-U, fourth International Conference on Database Theory (ICDT '92).

[7] J. Väänänen, The logic of approximate dependence, in: C. Başkent, L. S. Moss, R. Ramanujam (Eds.), Rohit Parikh on Logic, Language and Society, Springer International Publishing, Cham, 2017, pp. 227–234. doi:10.1007/978-3-319-47843-2_12.

[8] Y. Huhtala, J. Kärkkäinen, P. Porkka, H. Toivonen, Tane: An efficient algorithm for discovering functional and approximate dependencies, The Computer Journal 42 (1999) 100–111. doi:10.1093/comjnl/42.2.100.

[9] M. Hannula, J. Kontinen, F. Yang, Logics in Team Semantics, 2026. Unpublished manuscript.

[10] P. Galliani, J. Väänänen, On dependence logic, in: A. Baltag, S. Smets (Eds.), Johan van Benthem on Logic and Information Dynamics, Springer International Publishing, Cham, 2014, pp. 101–119. doi:10.1007/978-3-319-06025-5_4.

[11] M. Häggblom, Axiomatizing variants of approximate inclusion, in: A.-Y. Turhan, J. Virtema (Eds.), Foundations of Information and Knowledge Systems, Springer Nature Switzerland, Cham, 2026, pp. 341–346.

[12] J. Kontinen, Coherence and computational complexity of quantifier-free dependence logic formulas, Stud Logica 101 (2013) 267–291. doi:10.1007/S11225-013-9481-8.

[13] A. Durand, M. Hannula, J. Kontinen, A. Meier, J. Virtema, Approximation and dependence via multiteam semantics, Ann. Math. Artif. Intell. 83 (2018) 297–320. doi:10.1007/S10472-017-9568-4.

[14] S. S. Cosmadakis, P. C. Kanellakis, M. Y. Vardi, Polynomial-time implication problems for unary inclusion dependencies, J. ACM 37 (1990) 15–46. doi:10.1145/78935.78937.

[15] M. Hirvonen, The implication problem for functional dependencies and variants of marginal distribution equivalences, ACM Trans. Comput. Logic 25 (2024). doi:10.1145/3677120.

[16] T. Barlag, M. Hannula, J. Kontinen, N. Pardal, J. Virtema, Locally consistent k-relations: Entailment and axioms of functional dependence, 2026. arXiv:2505.11057.

[17] T. J. Green, G. Karvounarakis, V. Tannen, Provenance semirings, in: Proceedings of the Twenty-Sixth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS '07, Association for Computing Machinery, New York, NY, USA, 2007, p. 31–40. doi:10.1145/1265530.1265535.

[18] M. Hannula, Conditional Independence on Semiring Relations, in: G. Cormode, M. Shekelyan (Eds.), 27th International Conference on Database Theory (ICDT 2024), volume 290 of *Leibniz International Proceedings in Informatics (LIPIcs)*, Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl, Germany, 2024, pp. 20:1–20:20. doi:10.4230/LIPIcs.ICDT.2024.20.

[19] M. Hirvonen, Implication problems over positive semirings, in: Foundations of Information and Knowledge Systems: 14th International Symposium, FoIKS 2026, Hanover, Germany, March 23–26, 2026, Proceedings, Springer-Verlag, Berlin, Heidelberg, 2026, p. 29–47. doi:10.1007/978-3-032-21540-6_3.

[20] M. R. Garey, D. S. Johnson, L. J. Stockmeyer, Some simplified NP-complete graph problems, Theor. Comput. Sci. 1 (1976) 237–267.

[21] N. Fröhlich, P. G. Kolaitis, A. Meier, Disjunctions of two dependence atoms, in: S. Guerrini, B. König (Eds.), 34th EACSL Annual Conference on Computer Science Logic, CSL 2026, Paris, France, February 23-28, 2026, LIPIcs, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2026, pp. 10:1–10:21. doi:10.4230/LIPICS.CSL.2026.10.

Table 3
Types of error for v_1, v_2 and v_3. We denote by • any non zero value.

type	x	v_1	v_2	v_3
$\mathcal{A}$	0	•	0	0
$\mathcal{B}$	0	•	•	0
$\mathcal{C}$	0	•	•	•
$\mathcal{D}$	0	•	0	•
$\mathcal{E}$	0	0	0	•
$\mathcal{F}$	0	0	•	•
$\mathcal{G}$	0	0	•	0

A. Proofs

The appendix includes the proofs omitted from the main part of the paper.

Proposition 3.2. *Let $\Sigma = \{=_0(x), =_{\frac{1}{8}}(x, v_1v_2), =_{\frac{2}{8}}(x, v_1v_3), =_{\frac{2}{8}}(v_1, y), =_0(v_2v_3, y)\}$ be a set of approximate atoms. If $X \models \Sigma$ for any team X, then $X \models =_{\frac{5}{16}}(x, y)$.*

Proof. Fix any team $X \models \Sigma$. Notice that there are seven different types of 'error' assignments in X for variables v_1, v_2 and v_3, i. e., assignments which need to be removed to make $=_{\frac{1}{8}}(x, v_1v_2)$ or $=_{\frac{2}{8}}(x, v_1v_3)$ true. These 'error types' are depicted in Table 3 and labelled $\mathcal{A}$ to $\mathcal{G}$. Note that each type corresponds to a non-empty subset of the set $\{v_1, v_2, v_3\}$. We denote by a the ratio of error type $\mathcal{A}$ occurring in X, i. e.,

$$a = \frac{|\{s \in X \mid s \text{ is of error type } \mathcal{A}\}|}{|X|}.$$

Likewise define the ratios b to g.

We have two constraints on these ratios given by Σ:

$$a + b + c + d + f + g \leq \frac{1}{8}$$

the ratio of assignments which have to be removed for $=_{\frac{1}{8}}(x, v_1v_2)$, and

$$a + b + c + d + e + f \leq \frac{2}{8}.$$

the ratio of assignments which have to be removed for $=_{\frac{2}{8}}(x, v_1v_3)$.

We now show, that $p = \frac{5}{16}$ is the smallest ratio to guarantee $X \models =_p(x, y)$. Here, there are two ways to show $X \models =_p(x, y)$. First, via $=_q(x, v_2v_3)$ and $=_0(v_2v_3, y)$ with unknown ratio $q = b + c + d + e + f + g$. Second, via $=_r(x, v_1)$ and $=_{\frac{2}{8}}(v_1, y)$ with unknown ratio $r = a + b + c + d$. With transitivity it follows that $p = \min(q + 0, r + \frac{2}{8})$. We need to show that $p \leq \frac{5}{16}$ for any X, i. e., we want to show the maximum value p can take is less than or equal to $\frac{5}{16}$. Thus, we assume $q = r + \frac{2}{8}$ as the worst case for p and try to find the maximum. If we plug in the lower bounds we get

$$b + c + d + e + f + g = a + b + c + d + \frac{2}{8}$$

which implies $e + f + g = \frac{2}{8} + a$. From the first constraint, it follows that $f + g \leq \frac{1}{8} - r$. Thus $e \geq \frac{1}{8} + r + a$. From the second constraint, it follows that $e \leq \frac{2}{8} - r - f$. Together we have

$$\frac{2}{8} - r - f \geq \frac{1}{8} + r + a$$

$$\frac{1}{8} - a - f \geq 2r$$
$$\frac{1}{16} - \frac{a+f}{2} \geq r$$

Since we want to maximize r, assume $a = f = 0$ and $r = \frac{1}{16}$. So we get $p = \frac{2}{8} + \frac{1}{16} = \frac{5}{16}$ as desired. Therefore any team X that satisfies Σ must also satisfy $=_{\frac{5}{16}}(x,y)$. □

Theorem 4.1. *Let $\Sigma \cup \{=_p(x,y)\}$ be a finite set of unary and 1-constancy rational approximate dependence atoms. If $\Sigma \models =_p(x,y)$, then $\Sigma \vdash_{\mathbf{A}^1} =_p(x,y)$.*

Proof. Let D_0 be the set of variables in $\Sigma \cup \{=_p(x,y)\}$, and let z_0 be a variable that does not occur in any of the atoms in the set. We write $\vdash$ for a derivation using rules $A1^1$, $A2^1$, $A3^1$, $A6^1$ and $A7^1$. Suppose that $\Sigma \nvdash =_p(x,y)$. By $A1^1$ and $A7^1$, we can derive $\vdash =_0(x,x) \vdash =_p(x,x)$ and by $A2^1$ we can derive $\vdash =_1(x,y)$, so it follows that y is different from x and that $p < 1$.

For each variable u in D_0, define $d(u)$ to be the smallest rational number for which $\Sigma \vdash =_{d(u)}(x,u)$, which is guaranteed to exist since Σ is finite and $\vdash =_1(x,u)$ always holds by $A2^1$ (also in the case that x is the empty sequence).

Let $n-1$ be the least common denominator for the approximations appearing in $\Sigma \cup \{=_p(x,y)\}$. For each approximation q, let q' be such that $q = \frac{q'}{n-1}$, and similarly, we write $d(u)'$ whenever $d(u) = \frac{d(u)'}{n-1}$, etc. We build the counterexample team $X = \{s_0, \dots s_{n-1}\}$ such that for each variable $w \in D_0$,

$$s_i(w) = \begin{cases} i & \text{if } i \leq d(w)', \\ d(w)' & \text{otherwise.} \end{cases}$$

Additionally, let $s_i(z_0) = i$ for all $i \in \{0, \dots, n-1\}$.

The team is constructed such that for $w \in D_0$, $|X[w]| = d(w)' + 1$, see Table 2 for an example. As a consequence, we have the equivalences:

$$X \models =_q(u,v) \text{ iff } d(v)' - d(u)' \leq qn \text{ iff } |X[v] \setminus X[u]| \leq qn. \tag{$*$}$$

Now, let us show that $X \not\models =_p(x,y)$. Observe that $\Sigma \vdash =_0(x,x)$ by $A1^1$, so $X[x] = \{0\}$ if x is a nonempty sequence and otherwise $\emptyset$, so $|X[x]| \leq 1$. Furthermore, since $\Sigma \nvdash =_p(x,y)$, we have by construction of X that $|X[y]| \geq p' + 2$, hence $|X[y] \setminus X[x]| \geq p' + 1$. Now,

$$\frac{|X[y] \setminus X[x]|}{n} \geq \frac{p'+1}{n} > \frac{p'}{n-1} = p,$$

thus $X \not\models =_p(x,y)$ follows by $(*)$.

It remains to show that for each $=_r(u,v) \in \Sigma$, $X \models =_r(u,v)$. By construction of the team, $|X[u] \setminus X[x]| = d(u)'$. By rule $A6^1$ we have that $\Sigma \vdash =_{d(u)+r}(x,v)$, hence $|X[v] \setminus X[x]| \leq d(u)' + r'$. Now

$$|X[v] \setminus X[u]| \leq (d(u)' + r') - d(u)' = r' = r(n-1) < rn,$$

from which $X \models =_r(u,v)$ follows by $(*)$.

If we want to avoid the dummy variable z_0, identify the smallest k for which $s_k \in X$ and s_k, s_l are the same assignments restricted to the variables in D_0 whenever $k < l \leq n-1$. For each such s_l, replace it with the assignment s_l^a such that $s_l^a(w) = a_l$ for all $w \in D_0$, where a_l is a fresh value. Let Y be the team $\{s_0, \dots, s_k, s_{k+1}^a, \dots s_{n-1}^a\}$. Checking that Y is a counterexample team is similar to the proof for X. □

Theorem 5.1. *The model checking problem for $=_{\frac{3}{13}}(u,v) \vee =_0(x,y)$ is* NP*-complete.*

Proof. Membership in NP is quite obvious. A non-deterministic machine can guess the split into X_1 and X_2 for the disjunction and additionally guess the removed $\frac{3}{13}$ assignments $Y_1 \subseteq X_1$. Then the only thing left to do is to check whether $X_1 \setminus Y_1 \models =(u, v)$ and $X_2 \models =(x, y)$ is true.

We continue with showing NP-hardness. To this end, we present a reduction from 3SAT. This reduction will use a similar construction as the reduction to the decision problem of Max2SAT presented in [20].

Let ϕ be an instance of 3SAT, i. e., a set of clauses $(a_i \vee b_i \vee c_i)$ for $i \in I$, where each a_i, b_i and c_i corresponds to a variable or its negation. First, split each clause $(a_i \vee b_i \vee c_i)$ into four subclauses of size 1 and six subclauses of size 2:

$$(a_i), (b_i), (c_i), (d_i), (\bar{a}_i \vee \bar{b}_i), (\bar{a}_i \vee \bar{c}_i), (\bar{b}_i \vee \bar{b}_i), (a_i \vee \bar{d}_i), (b_i \vee \bar{d}_i), (c_i \vee \bar{d}_i).$$

The variable d_i is a fresh variable introduced for each clause. As noted in [20] exactly seven of these ten clauses can be true when the original clause was satisfied, while at most six can be true if a_i, b_i and c_i are false.

We now construct a team X_i such that the clause is satisfied if and only if a split between 17 of 20 assignments is possible. Let $p\colon \mathrm{Lit} \to \{0, 1\}$ be the function determining the parity of literals; while $\bar{p}$ denotes the opposite, i.e., $\bar{p}(x) = 1 - p(x)$ for all $x \in \mathrm{Lit}$. Now, the team X_i is as depicted in Table 4. The full team of this reduction is then the union of all X_i for $i \in I$, i.e. $X = \bigcup_{i \in I} X_i$.

Correctness follows from [20] in a mostly straightforward manner. First, observe that $=_0(x, y)$ corresponds to assignments of the variables. That is, in the split $X_2 \models =_0(x, y)$, each variable has to agree on its parity. Second, the split of $=_{\frac{3}{13}}(u, v)$ can take one assignment from each subclause without considering the approximation. Taking the approximation into account, this side of the split can take three additional assignments for a total of 13. Ordinarily, these three assignments would violate the dependence atom; however, they can be removed using the approximation. Thus, if seven of the ten subclauses are true by an assignment team, X_i can be split. If this is true for all X_i, then this also holds for X. In contrast, if only six assignments are true (as is the case for the non-satisfying assignment of the clause), then the team X_i cannot be split, and therefore also X cannot be split. □

Theorem 5.2. *The model checking problem for* $=_{\frac{1}{2}}(x, y)$ *is* LOGSPACE*-hard.*

Proof. Define $X^{\leftrightarrows}$ as the team that swaps and renames the values of x and y in every assignment of X. For example, if $s(x) = 0, s(y) = 1$ is an assignment in X, then $X^{\leftrightarrows}$ contains $\bar{s}(x) = \bar{1}, \bar{s}(y) = \bar{0}$ as an assignment.

Let X be a team over $\{x, y\}$. We reduce from the model checking problem for $=(x, y) \vee =(y, x)$, which is LOGSPACE-complete [21], by mapping X to $X \cup X^{\leftrightarrows}$. It is easy to see that this can be done by a first-order reduction.

Table 4
Team X_i given a clause over variables $\{a, b, c\}$, with $p(x)$ the parity of x in the clause; and $\bar{p}$ the opposite.

u	v	x	y
i_0	0	a	$p(a)$
i_0	1	a	$p(a)$
i_1	0	b	$p(b)$
i_1	1	b	$p(b)$
i_2	0	c	$p(c)$
i_2	1	c	$p(c)$
i_3	0	d_i	1
i_3	1	d_i	1

u	v	x	y
i_4	0	a	$\bar{p}(a)$
i_4	1	b	$\bar{p}(b)$
i_5	0	a	$\bar{p}(a)$
i_5	1	c	$\bar{p}(c)$
i_6	0	b	$\bar{p}(b)$
i_6	1	c	$\bar{p}(c)$

u	v	x	y
i_7	0	a	$p(a)$
i_7	1	d_i	0
i_8	0	b	$p(b)$
i_8	1	d_i	0
i_9	0	c	$p(c)$
i_9	1	d_i	0

Assume $X \models =(x,y) \vee =(y,x)$. Then there exists a split $X = X_1 \cup X_2$ such that $X_1 \models =(x,y)$ and $X_2 \models =(y,x)$. The same split also witnesses satisfaction for $X^{\leftrightarrows}$, when swapping the teams, i.e., $X_2^{\leftrightarrows} \models =(x,y)$ and $X_1^{\leftrightarrows} \models =(y,x)$. Since the values in X and $X^{\leftrightarrows}$ are disjoint $X_1 \cup X_2^{\leftrightarrows} \models =(x,y)$. Now, for the size of these subteams we have that

$$
\begin{aligned}
|X_1| + |X_2| &\geq |X| && \\
|X_1| + |X_2^{\leftrightarrows}| &\geq |X| && (|X_2| = |X_2^{\leftrightarrows}|) \\
|X_1 \cup X_2^{\leftrightarrows}| &\geq |X| && (X_1 \cap X_2^{\leftrightarrows} = \emptyset) \\
|X_1 \cup X_2^{\leftrightarrows}| &\geq \frac{1}{2} \cdot (|X| + |X^{\leftrightarrows}|) && (|X| = |X^{\leftrightarrows}|) \\
|X_1 \cup X_2^{\leftrightarrows}| &\geq \frac{1}{2} \cdot |X \cup X^{\leftrightarrows}| && (X \cap X^{\leftrightarrows} = \emptyset)
\end{aligned}
$$

Therefore $X \cup X^{\leftrightarrows} \models =_{\frac{1}{2}}(x,y)$.

For the other direction assume $X \not\models =(x,y) \vee =(y,x)$. Then for all subteams $X_1, X_2 \subseteq X$ with $X_1 \models =(x,y)$ and $X_2 \models =(y,x)$ we have that $X \setminus (X_1 \cup X_2) \neq \emptyset$. Therefore $|X_1| + |X_2| < |X|$. The same is true for $X^{\leftrightarrows}, X_1^{\leftrightarrows}$ and $X_2^{\leftrightarrows}$. Thus

$$
\begin{aligned}
|X_1| + |X_2| + |X_1^{\leftrightarrows}| + |X_2^{\leftrightarrows}| &< |X| + |X^{\leftrightarrows}| \\
2|X_1| + 2|X_2^{\leftrightarrows}| &< |X| + |X^{\leftrightarrows}| \\
|X_1| + |X_2^{\leftrightarrows}| &< \frac{1}{2}(|X| + |X^{\leftrightarrows}|) \\
|X_1 \cup X_2^{\leftrightarrows}| &< \frac{1}{2}|X \cup X^{\leftrightarrows}|
\end{aligned}
$$

holds and implies $X \cup X^{\leftrightarrows} \not\models =_{\frac{1}{2}}(x,y)$. □