Research Article

AND DATA SCIENCES
BULLETIN OF COMPUTER

Modular QBF Compilations for Implication, Abduction, and Repair in Team Semantics

Fei Yu

College of Information and Intelligent, Hunan Agricultural University, Changsha, China, 410128

Abstract

Recent work has shown that a broad spectrum of reasoning tasks in team semantics—notably model checking, counting, and enumeration—admits modular reductions to satisfiability in carefully controlled propositional fragments, yielding refined tractability frontiers. Building on that compilation paradigm, this paper studies three reasoning tasks that are central in knowledge representation and database practice but are not captured by model checking alone: (i) *implication* between dependency constraints, (ii) *abduction* of missing tuples or assignments required to satisfy a constraint, and (iii) *repair* via minimal modifications under block-aware decompositions. We present a uniform translation of these tasks for a wide class of team-based logics into *quantified Boolean formulas* (QBF). The translation is modular with respect to team connectives and dependency atoms, mirroring the syntactic modularity of team semantics. It yields immediate complexity upper bounds by landing in low alternation levels and syntactic fragments of QBF, and it isolates refined low-complexity subcases via Horn-like and 2-CNF-like QBF backbones. Conceptually, the results establish QBF as a natural "second compilation target" for team semantics whenever reasoning quantifies over teams, repairs, or explanations rather than merely verifying satisfaction.

Keywords: team semantics, quantified Boolean formulas, implication, abduction, database repair, modular compilation

1. Introduction

Team semantics is a now-standard semantic framework for logics of dependence and imperfect information [1]. In contrast to classical Tarskian semantics, where a formula is evaluated at a single assignment, team semantics evaluates formulas over *sets of assignments* (teams) [1, 2]. This change is not merely cosmetic: it makes it possible to express, within the object language, constraints that are inherently *relational* across multiple assignments, such as functional dependence, inclusion, exclusion, anonymity, and (conditional) independence [3]. From the perspective of database theory, a team can be viewed as a finite relation over a fixed set of attributes (variables), and dependency atoms correspond closely to integrity constraints and information-flow policies [4]. From the perspective of knowledge representation, teams provide a natural semantics for partial information, uncertainty, and multi-world reasoning, where a set of compatible assignments represents the epistemic state of an agent or system.

Corresponding author: Fei Yu (feiyu2@proton.me)

BCDS 2025, Vol. 6(4), 1-21. DOI: 10.71448/bcds2564-1

The move from single assignments to teams also changes the computational landscape in a characteristic way [6, 7]. Many fragments behave classically on literals and conjunction, but disjunction and quantification become *team-transforming* operations: disjunction splits a team into subteams, and quantifiers extend a team by adding new values for variables. These operations interact non-trivially with dependency atoms, and this interaction is precisely where intractability often emerges [6, 7]. In particular, even syntactically weak fragments can become computationally hard once unrestricted team disjunction is available alongside expressive dependencies, whereas carefully controlled fragments admit efficient algorithms. A major theme in the area is therefore to identify *structural reasons* for tractability and to develop techniques that make these tractable islands usable in practice.

A recent and influential line of work approaches this problem through *compilation*. Instead of reasoning about team semantics directly, one translates the semantic conditions induced by a team formula into propositional constraints, in such a way that satisfiability in a suitably chosen propositional fragment mirrors the original reasoning task [8]. This has produced a modular toolkit in which the contribution of each connective and each dependency atom is mapped to clauses in fragments such as dual-Horn, Horn, 2-CNF, implicative 2-CNF, and a special NL-decidable fragment (1-EDH). Beyond yielding refined complexity classifications, the compilation viewpoint has a clear methodological advantage: once a fragment compiles into a well-understood propositional class, one can immediately leverage decades of progress in SAT solving, preprocessing, and certificate generation. In this paper we build on that paradigm, but we target a different family of questions: those in which the reasoning task itself ranges *over teams* or *over modifications of teams*, rather than simply checking whether a given team satisfies a formula.

Such tasks arise naturally in applications. In databases, integrity constraints are rarely used only for validation; they are also used for *constraint implication* (does one set of constraints logically enforce another?), for *data completion* (can missing tuples be added so that constraints are satisfied?), and for *repair* (can a dataset be minimally modified to restore consistency?) [8]. In knowledge representation and diagnosis, one is often given partial evidence and seeks a small explanation that makes a theory consistent with observations, or a minimal set of changes that restores satisfaction. Team semantics provides a uniform language for stating the underlying constraints, but answering these meta-level questions is algorithmically more demanding because it requires quantification over candidate teams, completions, or repairs [9].

We therefore study three tasks that are central in practice and natural in team-semantics settings, yet not captured by model checking alone. The first is *implication*: given constraints φ and ψ, does φ entail ψ for *all* teams (optionally relative to a fixed structure)? The second is *abduction* (or completion): given a partial team X and a constraint φ, can we add a bounded explanation set E such that $X \cup E \models \varphi$? The third is *repair*: given a team X and a constraint φ, can we delete or reassign a small (or bounded) set of tuples to obtain a team X' that satisfies φ? In many decomposed settings one also requires that repairs respect semantic partitioning principles, for example *block disjunction* constraints that enforce that splitting or modification proceeds blockwise along a designated variable, reflecting locality, grouping, or operational constraints.

A key observation is that all three tasks require quantification over teams or team modifications. Under the compilation view, a team is represented by a family of Boolean membership variables, and model checking corresponds to fixing these variables according to the input team and asking for satisfiability of the compiled constraints [10]. Implication, abduction, and repair, in contrast, require reasoning of the form for all teams X" or there exists a completion/repair X'," i.e., they quantify

over those membership variables themselves. This shift makes quantified Boolean formulas (QBF) the natural compilation target. QBF not only provides a direct formalism for expressing universal and existential quantification over candidate teams, but it also offers a mature solver ecosystem and a well-developed theory connecting alternation depth and matrix fragments to complexity [11].

Our main contribution is a *modular compilation from team reasoning to QBF* that lifts the earlier SAT-based modularity to reasoning tasks with team quantification. The compilation is syntax-directed and preserves the modular character of team semantics: connectives such as conjunction, (split) disjunction, and quantifiers contribute predictable constraint patterns; dependency atoms contribute local clause schemata; and the reasoning task determines the QBF prefix (e.g., $\forall$ for implication, $\exists$ for abduction/repair, and mixed prefixes when both appear). This yields three immediate payoffs. First, we obtain a uniform representation of implication, bounded abduction, and bounded repair as QBF instances built from the same underlying compilation components. Second, we obtain transparent alternation bounds that separate the intrinsic difficulty contributed by the *task* (universal vs. existential team reasoning) from that contributed by the *logic* (which determines the matrix fragment). Third, we identify low-complexity subcases and solver-relevant structured instances when the compiled QBF falls into low-alternation and structurally restricted forms, notably Horn-like and 2-CNF-like matrices augmented by small, well-behaved cardinality or block-selection interfaces [11, 12].

The paper positions QBF as a natural "second compilation target" for team semantics whenever the question is not merely whether a fixed team satisfies a constraint, but whether there exists a suitable completion or repair, or whether a constraint universally implies another. This perspective extends the compilation toolbox, clarifies complexity drivers, and opens a path to solver-based workflows for explanation and repair problems stated in team-based languages.

2. Preliminaries

2.1. Teams and team semantics

Let $\mathcal{A}$ be a finite relational structure with domain A. We fix a finite set of variables $V = \{x_1, \ldots, x_r\}$ and write A^V for the set of all assignments $s : V \to A$. A *team* over V is a finite set $X \subseteq A^V$ of assignments. Intuitively, X represents a collection of compatible "possible worlds" or database rows over the attributes in V. In particular, when V is ordered as $(x_1, \ldots, x_r)$, each assignment $s \in X$ corresponds to the tuple

$$t_s := (s(x_1), \ldots, s(x_r)) \in A^r,$$

and thus a team can be identified with a finite relation $X \subseteq A^r$. This identification is convenient for compilation, where we will represent membership of tuples in X by propositional variables indexed by elements of A^r.

Satisfaction in team semantics is written $\mathcal{A} \models_X \varphi$ and is defined inductively over the structure of φ. We assume the standard *lax* (a.k.a. "Hodges") team semantics, which is the default semantics in most computational studies and has clean closure properties. For first-order literals α (including relational atoms and their negations under FO-negation), team semantics reduces to pointwise satisfaction:

$$\mathcal{A} \models_X \alpha \quad \text{iff} \quad \forall s \in X \, (\mathcal{A} \models_s \alpha),$$

in the ordinary Tarskian sense. Conjunction is classical:

$$\mathcal{A} \models_X (\varphi \wedge \psi) \quad \text{iff} \quad \mathcal{A} \models_X \varphi \text{ and } \mathcal{A} \models_X \psi.$$

Disjunction is *splitting*: $\mathcal{A} \models_X (\varphi \vee \psi)$ holds if the team can be partitioned into subteams that satisfy each disjunct,

$$\mathcal{A} \models_X (\varphi \vee \psi) \quad \text{iff} \quad \exists Y, Z \subseteq X \, (Y \cup Z = X \ \wedge \ \mathcal{A} \models_Y \varphi \ \wedge \ \mathcal{A} \models_Z \psi).$$

This splitting condition is the main source of non-classical behavior and is central to the compilation approach, because it introduces existentially chosen intermediate teams Y and Z.

Quantifiers act by *team extension*. For the existential quantifier, each assignment may be extended by choosing at least one witness value. Formally, for $F : X \to \mathcal{P}(A) \setminus \{\emptyset\}$ define the team

$$X[F/x] := \{\, s[a/x] \ : \ s \in X, \ a \in F(s) \,\},$$

where $s[a/x]$ is the assignment that agrees with s except mapping x to a. Then

$$\mathcal{A} \models_X \exists x \, \varphi \quad \text{iff} \quad \exists F : X \to \mathcal{P}(A) \setminus \{\emptyset\} \ \ \mathcal{A} \models_{X[F/x]} \varphi.$$

For the universal quantifier, extension is uniform:

$$\mathcal{A} \models_X \forall x \, \varphi \quad \text{iff} \quad \mathcal{A} \models_{X[A/x]} \varphi, \qquad X[A/x] := \{\, s[a/x] \ : \ s \in X, \ a \in A \,\}.$$

These definitions emphasize that reasoning about $\exists$ and $\vee$ naturally introduces auxiliary teams, which later become existentially quantified Boolean variables in our compilation.

Finally, team semantics is augmented with *dependency atoms*, which impose constraints on the team viewed as a relation. We will work with the standard family that frequently appears in applications and in complexity classifications: functional dependence $= (\vec{x}, \vec{y})$, inclusion $\vec{x} \subseteq \vec{y}$, exclusion $\vec{x} \mid \vec{y}$, anonymity $\Upsilon \vec{x}$, constancy $= (\vec{x})$, and (conditional) independence $\vec{y} \perp_{\vec{x}} \vec{z}$. We do not need the full formal definitions here; it suffices to note that each atom is evaluated directly on X (as a relation) and can be compiled into local propositional constraints over membership variables. In particular, several of these atoms yield Horn- or dual-Horn-like constraints when expressed over Boolean encodings of tuple membership, which is precisely what enables refined tractability results.

2.2. Reasoning tasks beyond model checking

Let $\mathcal{L}$ be a fixed class of formulas interpreted under team semantics, and let $\mathcal{A}$ be a fixed finite structure. The classical *model checking* problem takes $(\mathcal{A}, X, \varphi)$ as input and asks whether $\mathcal{A} \models_X \varphi$. In contrast, the present paper focuses on tasks in which the input team is not merely checked, but is treated as a variable object that we quantify over or modify.

Definition 1 (Implication over teams)**.** *Given $\varphi, \psi \in \mathcal{L}$ with $\mathrm{Fr}(\varphi) \cup \mathrm{Fr}(\psi) \subseteq V$, the implication problem asks whether*

$$\forall X \subseteq A^{|V|} \, (\mathcal{A} \models_X \varphi \Rightarrow \mathcal{A} \models_X \psi).$$

Implication is a natural "static analysis" question for constraints: it asks whether enforcing φ automatically enforces ψ on *every* admissible dataset/team. In database language, it corresponds to entailment between integrity constraints relative to $\mathcal{A}$. The universal quantification over X is the key feature that forces a move beyond SAT-based compilation.

Definition 2 (Bounded abduction (superteam existence))**.** *Given $\varphi \in \mathcal{L}$, a base team $X \subseteq A^{|V|}$, and a bound k, decide whether there exists $E \subseteq A^{|V|}$ with $|E| \leq k$ such that*

$$\mathcal{A} \models_{X \cup E} \varphi.$$

Here E can be read as an *explanation* or *completion* set: we ask whether φ can be made true by adding at most k missing tuples/assignments. This captures a common scenario in incomplete data and diagnosis: a constraint fails on the observed team X, but might hold after adding a small number of plausible missing observations.

Definition 3 (Deletion repair (subteam existence))**.** *Given $\varphi \in \mathcal{L}$, a team X, and a bound k, decide whether there exists $X' \subseteq X$ with $|X \setminus X'| \leq k$ such that $\mathcal{A} \models_{X'} \varphi$.*

Deletion repair asks whether a constraint can be restored by removing at most k tuples. This is the canonical repair notion in inconsistent databases (tuple deletions), and it is also a natural robustness measure: if only a small number of assignments need to be removed, then φ is "almost" satisfied. In practice, one often also studies optimization variants (minimum deletions, lexicographic preferences, weighted costs), but bounded decision forms are sufficient for our compilation framework and can serve as subroutines in iterative optimization.

2.3. Quantified Boolean formulas (QBF)

A quantified Boolean formula (QBF) is a formula of the form [5]

$$Q_1 \vec{p}_1 \; Q_2 \vec{p}_2 \; \cdots \; Q_m \vec{p}_m \; \Phi,$$

where each $Q_i \in \{\exists, \forall\}$, each $\vec{p}_i$ is a block of Boolean variables, and Φ is a quantifier-free propositional matrix (typically in CNF). The *alternation depth* (the number of switches between $\exists$ and $\forall$ in the prefix) controls the position of the problem in the polynomial hierarchy for polynomial-size encodings. At the same time, the syntactic shape of Φ—for example, whether it is Horn, dual-Horn, or 2-CNF—often determines whether specialized procedures exist and whether the resulting instances are practically solvable at scale. This aligns closely with the compilation philosophy: we aim to keep the matrix in a structured fragment whenever the underlying team logic fragment allows it, while letting the quantifier prefix reflect only the reasoning task.

3. Modular QBF compilation for team reasoning

This section presents the central technical idea of the paper: a uniform, syntax-directed compilation that turns satisfaction in team semantics into propositional constraints, and then lifts this compilation to meta-reasoning tasks—implication, abduction, and repair—by placing the *team-encoding variables* under suitable QBF quantifiers. The guiding principle is that the *matrix* of the QBF is obtained by the same modular translation used for SAT-based model checking, while the *quantifier prefix* reflects only the reasoning task (universal quantification for entailment, existential quantification for completion/repair, and mixed quantification when both interact).

3.1. Team encodings as Boolean characteristic functions

Fix a team vocabulary $V = \{x_1, \ldots, x_r\}$ and a finite structure $\mathcal{A}$ with domain A. Under the standard identification of teams with relations, every team $X \subseteq A^r$ can be represented by its characteristic function. We therefore introduce a family of Boolean variables

$$T[t] \qquad \text{for each } t \in A^r,$$

where $T[t] = 1$ is intended to mean that the tuple t is a member of the team X (equivalently, the assignment associated with t is in X). This is the basic interface between team semantics and propositional reasoning: once teams are represented by membership variables, semantic conditions become constraints over these variables.

Team semantics is compositional but *team-transforming*. In particular, disjunction and existential quantification introduce auxiliary teams whose existence is witnessed by suitable splits and extensions. To mirror this structure, the compilation introduces fresh membership families for intermediate teams created along the syntax tree of a formula. Concretely, for each relevant subformula θ (and, when needed, for auxiliary teams naturally associated with θ), we introduce a family

$$T_\theta[\cdot]$$

intended to represent the team at which θ is evaluated during a successful semantic derivation. For example, for $\varphi \vee \psi$ we introduce team variables $T_\varphi[\cdot]$ and $T_\psi[\cdot]$ representing the two subteams witnessing the split; for $\exists x\, \varphi$ we introduce a team variable representing the extended team after choosing witnesses for x. The compilation then consists of local clause schemata that enforce the relationships dictated by team semantics, such as $X = Y \cup Z$ for disjunction, or closure conditions for universal extension.

In SAT-based model checking, the input team X is given, and one fixes the values of the $T[t]$ variables using unit clauses consistent with X. The compiled constraints are satisfiable exactly when $\mathcal{A} \models_X \varphi$. The crucial step in this paper is to generalize from *fixed* teams to *quantified* teams: in implication, abduction, and repair, the team itself (or a modification of it) is the unknown object we are solving for. In the membership-variable representation, this means that we no longer assign $T[t]$ by input, but instead quantify over $T[\cdot]$ (and over additional modification variables). This is precisely what makes QBF the appropriate target language.

3.2. The compiled matrix and the role of auxiliary witnesses

Let $\mathrm{Comp}_0(\varphi; T, U)$ denote the quantifier-free propositional matrix produced by the modular compilation of a team formula φ, where T is the membership family representing the *input* team, and U is the collection of all auxiliary variables introduced for intermediate teams, splits, and quantifier witnesses. The compilation is designed to satisfy the semantic correspondence:

$$\mathcal{A} \models_T \varphi \quad \Longleftrightarrow \quad \exists U\ \mathrm{Comp}_0(\varphi; T, U).$$

Intuitively, U encodes all existential semantic choices that arise when evaluating φ under lax team semantics: how to split the team for disjunction, which witness sets to choose for existential quantification, and any additional local witness choices needed by particular dependency atoms (depending on the atom and the chosen compilation scheme). When φ belongs to a syntactic fragment with favorable closure properties, these constraints can be expressed in restricted propositional classes (Horn, dual-Horn, 2-CNF, implicative 2-CNF, or, in certain weak grammars, the NL-decidable 1-EDH fragment). This is the main reason the compilation method yields refined tractability frontiers: the semantic operator determines a small constraint pattern, and the global syntactic fragment determines the overall matrix class.

The distinction between SAT compilation and QBF compilation is therefore entirely *external* to the matrix: SAT-based model checking fixes T and asks for satisfiability of $\exists U\, \mathrm{Comp}_0$, whereas our meta-reasoning tasks quantify over T or over derived membership families and ask validity/satisfiability of the resulting QBF.

3.3. QBF skeletons for implication, abduction, and repair

Once Comp_0 is available, the three target reasoning tasks can be expressed by simple QBF "skeletons" that wrap the same compiled matrix in different quantifier prefixes.

Implication $\varphi \Rightarrow \psi$ requires that *every* team that satisfies φ also satisfies ψ. In terms of membership variables and the compiled matrix, this can be stated as

$$\forall T \Big((\exists U_\varphi \, \mathrm{Comp}_0(\varphi; T, U_\varphi)) \to (\exists U_\psi \, \mathrm{Comp}_0(\psi; T, U_\psi)) \Big).$$

Equivalently, by prenexing and using that the witnesses for ψ do not occur in $\mathrm{Comp}_0(\varphi; \cdot)$, we obtain a $\forall\exists$-QBF with one alternation block:

$$\forall T \, \forall U_\varphi \, \exists U_\psi \, \Big(\mathrm{Comp}_0(\varphi; T, U_\varphi) \to \mathrm{Comp}_0(\psi; T, U_\psi) \Big),$$

or in standard disjunctive form,

$$\forall T \, \forall U_\varphi \, \exists U_\psi \, (\neg \mathrm{Comp}_0(\varphi; T, U_\varphi) \ \vee \ \mathrm{Comp}_0(\psi; T, U_\psi)).$$

The alternation remains inherent and conceptually clean: T is universally quantified because implication ranges over all teams, and the witness variables for φ move under the same universal block because entailment cannot be certified by *choosing* a convenient (possibly incorrect) antecedent witness. The only existential block is for witnesses of ψ, matching the semantic requirement that whenever $X \models \varphi$ holds (for some witness) then $X \models \psi$ must hold (for some witness). Importantly, the *matrix structure* is inherited from the fragments in which φ and ψ lie; thus, implication compiles to a low-alternation QBF whose hardness stems from the task-level universal quantification rather than from unstructured propositional content.

Bounded abduction asks whether a given base team T can be *extended* by adding at most k tuples so that φ becomes satisfied. We introduce explanation variables

$$E[t] \qquad (t \in A^r),$$

with intended meaning "t is added." The size bound $|E| \leq k$ is enforced by a standard cardinality encoding $\mathrm{Card}_{\leq k}(E)$. The abduced (augmented) team is represented by a derived membership family S with definitional equivalences

$$S[t] \leftrightarrow (T[t] \vee E[t]) \qquad (t \in A^r).$$

Then bounded abduction becomes

$$\exists E \exists S \exists U \, \Big(\mathrm{Card}_{\leq k}(E) \wedge \bigwedge_{t \in A^r} (S[t] \leftrightarrow (T[t] \vee E[t])) \wedge \mathrm{Comp}_0(\varphi; S, U) \Big).$$

Here the prefix is purely existential (once the observed base team T is fixed by input), and thus bounded abduction often compiles into SAT-like instances augmented by cardinality constraints. The key advantage of presenting it in this uniform framework is that the *semantic content* of φ still appears only through $\mathrm{Comp}_0(\varphi; \cdot)$, so Horn/dual-Horn/2-CNF structure can be preserved up to the relatively standard cardinality gadget.

Deletion repair asks whether one can obtain a satisfying subteam by removing at most k tuples. We introduce keep-variables

$$K[t] \qquad (t \in A^r),$$

intended to represent membership in the repaired team X'. Subteamhood is enforced by

$$K[t] \to T[t] \qquad (t \in A^r),$$

and the deletion bound is captured by a cardinality condition over deleted tuples, abstractly denoted $\mathrm{DelBound}_{\le k}(T, K)$, which can be implemented using a counter over indicators $(T[t] \wedge \neg K[t])$. The repair condition becomes

$$\exists K \exists U \Big(\bigwedge_{t \in A^r} (K[t] \to T[t]) \ \wedge \ \mathrm{DelBound}_{\le k}(T, K) \ \wedge \ \mathrm{Comp}_0(\varphi; K, U) \Big).$$

As with abduction, the prefix is existential given a fixed input team T, and the compiled matrix inherits the propositional fragment structure of φ aside from the standard cardinality interface.

These skeletons make explicit the paper's organizing message: the same compilation matrix Comp_0 is reused across tasks, and the only substantive change between model checking, implication, abduction, and repair is *where the membership variables live in the quantifier prefix.* In particular, implication necessarily introduces a universal block over the team encoding, while abduction and repair remain existential search problems unless additional universal requirements are imposed.

3.4. A modular compilation theorem for QBF

We now state the main meta-theorem underlying the framework. It formalizes that satisfaction in a broad class of team logics can be reduced to satisfiability of a compiled propositional matrix, and that this matrix is produced in a modular, syntax-directed manner that preserves membership in standard propositional fragments. The theorem also clarifies why the same compiled matrix can be embedded into the QBF skeletons above without losing these fragment guarantees.

Theorem 1 (Modular compilation to QBF)**.** *Let $\mathcal{L}$ be a team-based logic obtained by extending FO with a set of dependency atoms from*

$$\{=(\cdot,\cdot),\ \subseteq,\ |,\ \Upsilon,\ =(\cdot),\ \bot\},$$

and with team connectives and quantifiers from $\{\wedge, \vee, \exists, \forall, \vee^b_x\}$ under lax semantics. For every fixed $\varphi \in \mathcal{L}$ and finite structure $\mathcal{A}$ with domain A, there is a logspace-uniform construction of a propositional matrix $\mathrm{Comp}_0(\varphi; T, U)$ such that for every team encoding T over $V \supseteq \mathrm{Fr}(\varphi)$:

$$\mathcal{A} \models_T \varphi \quad \Longleftrightarrow \quad \exists U\ \mathrm{Comp}_0(\varphi; T, U).$$

Moreover, each syntactic construct of φ contributes matrix clauses in a predictable propositional fragment (Horn/dual-Horn/2-CNF/implicative 2-CNF, and in certain weak grammars, 1-EDH), and these fragment guarantees are preserved when embedding $\exists U\, \mathrm{Comp}_0(\varphi; T, U)$ into the QBF encodings of implication, bounded abduction, and deletion repair by quantifying over the appropriate team-encoding variables.

Proof. Fix a finite structure $\mathcal{A}$ with domain A and a finite variable set $V = \{x_1, \dots, x_r\}$ such that $\mathrm{Fr}(\varphi) \subseteq V$. We identify teams X over V with subsets of A^r via the tuple map $s \mapsto (s(x_1), \dots, s(x_r))$. A *team encoding* T is then a family of Boolean values $T[t] \in \{0, 1\}$ indexed by $t \in A^r$, interpreted as $t \in X$ iff $T[t] = 1$.

We define, by structural recursion on φ, a propositional matrix

$$\mathrm{Comp}_0(\varphi; T, U),$$

where T is the membership family for the input team and U is the union of all auxiliary variables introduced during compilation (membership families for intermediate teams and any additional witness variables). The compilation is *modular*: each syntactic constructor contributes a fixed pattern of clauses linking the input membership variables of the constructor to the membership variables of its immediate subformulas. We prove by induction on φ that for every team encoding T,

$$\mathcal{A} \models_T \varphi \quad \Longleftrightarrow \quad \exists U\, \mathrm{Comp}_0(\varphi; T, U).$$

Logspace-uniformity and fragment preservation are argued after the semantic equivalence.

For a tuple $t \in A^r$ we write $t[i]$ for its ith coordinate. If $x = x_i \in V$ and $a \in A$, define $t[a/x]$ to be the tuple obtained from t by replacing coordinate i by a. When $\vec{x}$ is a tuple of variables, $\pi_{\vec{x}}(t)$ denotes the projection of t to the coordinates of $\vec{x}$. We also use the abbreviation $\mathrm{In}_T(t)$ for the propositional literal $T[t]$.

For each subformula θ of φ we introduce a membership family $T_\theta[\cdot]$ representing the team on which θ is evaluated in a satisfying semantic derivation. The top-level input team is represented by T_φ, and we identify T_φ with the given T. Hence, compilation for φ produces constraints that enforce T_θ-families to witness the team semantics of θ.

Formally, we define $\mathrm{Comp}_0(\theta; T_\theta, U_\theta)$ for each θ such that U_θ collects all variables introduced in the compilation below θ. For the theorem statement, we take U as the disjoint union of all U_θ. We now specify the clause patterns for each syntactic case.

1) Let θ be a first-order literal (atomic or negated atomic in the FO sense) over V. For each tuple $t \in A^r$, the truth of θ under assignment t is decidable by inspecting $\mathcal{A}$ and the interpretation of relation symbols. Define a constant predicate $\mathrm{Sat}_\theta(t) \in \{0, 1\}$ such that $\mathrm{Sat}_\theta(t) = 1$ iff $\mathcal{A} \models_t \theta$ (Tarskian satisfaction at the assignment corresponding to t). We enforce pointwise satisfaction by

$$\mathrm{Comp}_0(\theta; T_\theta) \;:=\; \bigwedge_{t \in A^r:\ \mathrm{Sat}_\theta(t)=0} \neg T_\theta[t].$$

Thus, any tuple violating θ is forbidden from being in the team.

2) If $\theta = \alpha \wedge \beta$, team semantics requires $\mathcal{A} \models_X \alpha$ and $\mathcal{A} \models_X \beta$ on the same team. We simply reuse the same membership family for both subformulas:

$$\mathrm{Comp}_0(\alpha \wedge \beta; T_\theta, U_\alpha \cup U_\beta) := \mathrm{Comp}_0(\alpha; T_\theta, U_\alpha) \;\wedge\; \mathrm{Comp}_0(\beta; T_\theta, U_\beta).$$

3) If $\theta = \alpha \vee \beta$, lax semantics requires teams Y, Z with $Y \cup Z = X$ and $\mathcal{A} \models_Y \alpha$, $\mathcal{A} \models_Z \beta$. We introduce fresh membership families $T_\alpha[\cdot]$ and $T_\beta[\cdot]$ intended to encode Y and Z. We enforce $Y, Z \subseteq X$ and $X \subseteq Y \cup Z$ by the clause schemata, for all $t \in A^r$,

$$T_\alpha[t] \to T_\theta[t], \qquad T_\beta[t] \to T_\theta[t], \qquad T_\theta[t] \to (T_\alpha[t] \vee T_\beta[t]).$$

Then

$$\mathrm{Comp}_0(\alpha \vee \beta; T_\theta, U_\alpha \cup U_\beta \cup \{T_\alpha, T_\beta\}) := \Big(\bigwedge_{t \in A^r} \Psi_\vee(t) \Big) \;\wedge\; \mathrm{Comp}_0(\alpha; T_\alpha, U_\alpha) \;\wedge\; \mathrm{Comp}_0(\beta; T_\beta, U_\beta),$$

where $\Psi_\vee(t)$ abbreviates the three implications above.

4) If $\theta = \exists x\,\alpha$ where $x = x_i \in V$, lax semantics says that there exists a choice function $F : X \to \mathcal{P}(A) \setminus \{\emptyset\}$ such that $\mathcal{A} \models_{X[F/x]} \alpha$. We compile this by introducing a membership family $T_\alpha[\cdot]$ for the extended team and a family of *witness* variables $W[t, a]$ (for $t \in A^r$, $a \in A$) that indicate that a is chosen as a witness for x at base tuple t.

We enforce:

(i) Non-emptiness of witnesses:

$$T_\theta[t] \;\to\; \bigvee_{a \in A} W[t, a],$$

(ii) Witnesses are attached to base tuples and generate extension tuples:

$$W[t, a] \;\to\; T_\theta[t] \qquad \text{and} \qquad W[t, a] \;\to\; T_\alpha[t[a/x]],$$

(iii) Extension tuples originate from some base with a witness:

$$T_\alpha[u] \;\to\; \bigvee_{\substack{t \in A^r: \\ u = t[a/x] \text{ for some } a}} W[t, u[i]].$$

(Clauses (ii) and (iii) ensure that the compiled extension team is *exactly* an F-extension of the base team, i.e., it contains no tuples that are not generated from some $t \in X$.) Then we set

$$\mathrm{Comp}_0(\exists x\,\alpha; T_\theta,\; U_\alpha \cup \{T_\alpha, W\}) := \Big(\bigwedge_{t \in A^r} \Psi_\exists(t) \Big) \;\wedge\; \mathrm{Comp}_0(\alpha; T_\alpha, U_\alpha),$$

where $\Psi_\exists(t)$ denotes the conjunction of instances of (i)–(iii) relevant to t.

5) If $\theta = \forall x\,\alpha$, lax semantics requires $\mathcal{A} \models_{X[A/x]} \alpha$ where $X[A/x]$ contains *all* x-extensions of assignments in X. Introduce T_α for the extended team. For all $t \in A^r$ and all $a \in A$ enforce

$$T_\theta[t] \;\to\; T_\alpha[t[a/x]] \qquad \text{and} \qquad T_\alpha[u] \;\to\; \bigvee_{\substack{t \in A^r: \\ u = t[a/x] \text{ for some } a}} T_\theta[t].$$

The first implication enforces closure under all x-extensions; the second ensures every tuple in the extended team comes from some base tuple. Then define

$$\mathrm{Comp}_0(\forall x\,\alpha; T_\theta, U_\alpha \cup \{T_\alpha\}) := \Big(\bigwedge_{t,a} \Psi_\forall(t, a) \Big) \;\wedge\; \mathrm{Comp}_0(\alpha; T_\alpha, U_\alpha).$$

6) For $\theta = \alpha \;\vee^b_x\; \beta$ (block disjunction along variable x), the semantics requires a split $X = Y \cup Z$ such that for each value $a \in A$, either *all* tuples in X with $x = a$ go to Y or *all* go to Z. Introduce T_α, T_β as in ordinary disjunction, and additionally introduce block selector variables $B[a]$ for each $a \in A$ with intended meaning: if $B[a] = 1$ then the $x = a$ block is assigned to the α-branch; otherwise to the β-branch. Let $x = x_i$. Enforce as before that Y, Z form a cover of X and subsets of X, and additionally for all $t \in A^r$:

$$T_\theta[t] \wedge B[t[i]] \;\to\; T_\alpha[t], \qquad T_\theta[t] \wedge \neg B[t[i]] \;\to\; T_\beta[t].$$

These constraints ensure that block membership is consistent with a single selector for each x-value. Define Comp_0 for $\vee^b_x$ by conjoining these clauses with the compiled subformulas.

We now give clause schemata for the atoms listed in the theorem. Each schema is expressed solely in terms of the membership variables $T_\theta[\cdot]$ for the current subformula θ (no additional semantic witnesses are needed for the atoms listed, under the standard definitions).

7) Functional dependence $= (\vec{x}, \vec{y})$. Let θ be the atom $= (\vec{x}, \vec{y})$. Semantics: for all $s, s' \in X$, if $s(\vec{x}) = s'(\vec{x})$ then $s(\vec{y}) = s'(\vec{y})$. In tuple form, for all $t, t' \in A^r$,

$$T_\theta[t] \wedge T_\theta[t'] \wedge (\pi_{\vec{x}}(t) = \pi_{\vec{x}}(t')) \;\rightarrow\; (\pi_{\vec{y}}(t) = \pi_{\vec{y}}(t')).$$

We compile this by forbidding violating pairs: for all t, t' with $\pi_{\vec{x}}(t) = \pi_{\vec{x}}(t')$ but $\pi_{\vec{y}}(t) \neq \pi_{\vec{y}}(t')$, add the Horn clause

$$\neg T_\theta[t] \;\vee\; \neg T_\theta[t'].$$

8) Constancy $= (\vec{x})$. This is the special case $= (\emptyset, \vec{x})$: all tuples in the team must agree on $\vec{x}$. Thus for all t, t' with $\pi_{\vec{x}}(t) \neq \pi_{\vec{x}}(t')$ we add $\neg T_\theta[t] \vee \neg T_\theta[t']$.

9) Exclusion $\vec{x} \mid \vec{y}$. Semantics: no $s, s' \in X$ satisfy $s(\vec{x}) = s'(\vec{y})$. Thus for all t, t' with $\pi_{\vec{x}}(t) = \pi_{\vec{y}}(t')$ we add the Horn clause

$$\neg T_\theta[t] \;\vee\; \neg T_\theta[t'].$$

10) Inclusion $\vec{x} \subseteq \vec{y}$. Semantics: for each $s \in X$ there exists $s' \in X$ with $s(\vec{x}) = s'(\vec{y})$. For each $t \in A^r$, define the set

$$M_{\vec{x} \subseteq \vec{y}}(t) := \{\, t' \in A^r : \pi_{\vec{y}}(t') = \pi_{\vec{x}}(t) \,\}.$$

Then inclusion is:

$$T_\theta[t] \;\rightarrow \bigvee_{t' \in M_{\vec{x} \subseteq \vec{y}}(t)} T_\theta[t'].$$

We compile this as a dual-Horn clause for each t:

$$\neg T_\theta[t] \;\vee \bigvee_{t' \in M_{\vec{x} \subseteq \vec{y}}(t)} T_\theta[t'].$$

11) Anonymity $\Upsilon \vec{x}$. Under the standard definition, anonymity requires that for every assignment in the team and every alternative value for the anonymized variables, the corresponding tuple is also in the team (or, in equivalent formulations, that the projection on the non-anonymized variables forms complete blocks). This yields implications of the form $T_\theta[t] \rightarrow T_\theta[t']$ whenever t' agrees with t on the non-anonymized coordinates. Such implications are Horn (one positive head) and can also be written in dual-Horn normal form depending on the chosen polarity convention. Concretely, if $\vec{x}$ are the anonymized coordinates and $\vec{z} = V \setminus \vec{x}$ are the fixed coordinates, then for all t, t' with $\pi_{\vec{z}}(t) = \pi_{\vec{z}}(t')$ we add

$$T_\theta[t] \rightarrow T_\theta[t'] \quad \text{i.e.,} \quad \neg T_\theta[t] \vee T_\theta[t'].$$

12) Independence $\vec{y} \perp_{\vec{x}} \vec{z}$. Semantics: for all $s, s' \in X$ with $s(\vec{x}) = s'(\vec{x})$ there exists $s'' \in X$ such that $s''(\vec{x}\vec{y}) = s(\vec{x}\vec{y})$ and $s''(\vec{x}\vec{z}) = s'(\vec{x}\vec{z})$. Let $t, t' \in A^r$ with $\pi_{\vec{x}}(t) = \pi_{\vec{x}}(t')$. Define the "amalgamation" tuple $u = u(t, t')$ by setting $\pi_{\vec{x}\vec{y}}(u) = \pi_{\vec{x}\vec{y}}(t)$ and $\pi_{\vec{x}\vec{z}}(u) = \pi_{\vec{x}\vec{z}}(t')$. Then independence requires:

$$T_\theta[t] \wedge T_\theta[t'] \;\rightarrow\; T_\theta[u(t, t')].$$

We compile this as the Horn clause

$$\neg T_\theta[t] \;\vee\; \neg T_\theta[t'] \;\vee\; T_\theta[u(t, t')].$$

We prove the claimed equivalence by induction on the structure of φ.

Base case (literals and dependency atoms). If φ is an FO literal, then by construction $\mathrm{Comp}_0(\varphi; T)$ precisely forbids tuples t for which $\mathcal{A} \not\models_t \varphi$. Hence, for a team X encoded by T, $\exists U\, \mathrm{Comp}_0$ holds (with empty U) iff every $t \in X$ satisfies φ, i.e., $\mathcal{A} \models_X \varphi$.

If φ is a dependency atom, the compilation clauses are direct propositional restatements of the corresponding team condition. For dependence/constancy/exclusion, the clauses forbid exactly the violating pairs. For inclusion, each tuple in the team requires the existence of a matching tuple, which is encoded by a dual-Horn clause. For anonymity, closure of each block is encoded by Horn implications. For independence, each compatible pair requires the amalgamation tuple, encoded by a Horn 3-clause. In all cases, satisfaction of the atom by X is equivalent to the truth of the compiled clauses under the assignment that sets $T[t] = 1$ iff $t \in X$.

Inductive steps. Assume the equivalence holds for immediate subformulas of φ.

Conjunction. For $\varphi = \alpha \wedge \beta$, $\mathcal{A} \models_X \alpha \wedge \beta$ iff $\mathcal{A} \models_X \alpha$ and $\mathcal{A} \models_X \beta$. By the IH, each is equivalent to existence of witnesses U_α and U_β satisfying the compiled matrices on the same input membership family T. Thus the conjunction of the two compiled matrices is satisfiable iff both semantic conditions hold.

Disjunction. For $\varphi = \alpha \vee \beta$, $\mathcal{A} \models_X \alpha \vee \beta$ iff there exist $Y, Z \subseteq X$ with $Y \cup Z = X$ and $\mathcal{A} \models_Y \alpha$, $\mathcal{A} \models_Z \beta$.

- (*Soundness*) Suppose $\exists U\, \mathrm{Comp}_0(\alpha \vee \beta; T, U)$ holds. Then there exist truth assignments to the membership families T_α, T_β such that the split constraints hold. Define teams $Y := \{t : T_\alpha[t] = 1\}$ and $Z := \{t : T_\beta[t] = 1\}$. The split constraints ensure $Y, Z \subseteq X$ and $Y \cup Z = X$. Moreover, the compiled constraints include $\exists U_\alpha\, \mathrm{Comp}_0(\alpha; T_\alpha, U_\alpha)$ and similarly for β, so by IH, $\mathcal{A} \models_Y \alpha$ and $\mathcal{A} \models_Z \beta$, whence $\mathcal{A} \models_X \alpha \vee \beta$.

- (*Completeness*) Conversely, suppose $\mathcal{A} \models_X \alpha \vee \beta$. Choose witnessing teams Y, Z. Set $T_\alpha[t] = 1$ iff $t \in Y$ and $T_\beta[t] = 1$ iff $t \in Z$. The split constraints are satisfied by construction. By IH, there exist witnesses U_α, U_β satisfying the compiled matrices for α on T_α and for β on T_β. Together these assignments satisfy $\mathrm{Comp}_0(\alpha \vee \beta; T, U)$.

Existential quantifier. For $\varphi = \exists x\, \alpha$, $\mathcal{A} \models_X \exists x\, \alpha$ iff there exists $F : X \to \mathcal{P}(A) \setminus \{\emptyset\}$ such that $\mathcal{A} \models_{X[F/x]} \alpha$.

- (*Soundness*) Suppose $\exists U\, \mathrm{Comp}_0(\exists x\, \alpha; T, U)$ holds. Then there exist assignments to $W[t, a]$ and $T_\alpha[\cdot]$ satisfying constraints (i)–(iii). By (ii), $W[t, a] = 1$ implies $t \in X$ (i.e., $T[t] = 1$), so witnesses are chosen only for base tuples. Define F on X by $F(t) := \{a : W[t, a] = 1\}$ for $t \in X$. Constraint (i) ensures $F(t) \neq \emptyset$ whenever $t \in X$. Let $X' := \{u : T_\alpha[u] = 1\}$. The second part of (ii) yields $X[F/x] \subseteq X'$, and (iii) yields $X' \subseteq X[F/x]$; hence $X' = X[F/x]$. The compiled instance also contains $\exists U_\alpha\, \mathrm{Comp}_0(\alpha; T_\alpha, U_\alpha)$, so by IH, $\mathcal{A} \models_{X'} \alpha$. Hence $\mathcal{A} \models_X \exists x\, \alpha$.

- (*Completeness*) Conversely, suppose $\mathcal{A} \models_X \exists x\, \alpha$ with witnessing F and $X' = X[F/x]$. Set $T_\alpha[u] = 1$ iff $u \in X'$, and set $W[t, a] = 1$ iff $t \in X$ and $a \in F(t)$ (and $W[t, a] = 0$ otherwise). Then (i)–(iii) are satisfied: (i) holds because each $F(t)$ is nonempty, (ii) holds by construction, and (iii) holds because every $u \in X'$ has the form $t[a/x]$ for some $t \in X$ and $a \in F(t)$. By IH applied to α on X', choose witnesses U_α satisfying $\mathrm{Comp}_0(\alpha; T_\alpha, U_\alpha)$. Thus $\mathrm{Comp}_0(\exists x\, \alpha; T, U)$ is satisfiable.

Universal quantifier. For $\varphi = \forall x\,\alpha$, $\mathcal{A} \models_X \forall x\,\alpha$ iff $\mathcal{A} \models_{X[A/x]} \alpha$. Soundness and completeness follow as in the existential case, but without witness variables: the constraints enforce that T_α encodes exactly $X[A/x]$. Then apply IH to α.

Block disjunction. The argument is the same as for disjunction, except that the additional block selector variables $B[a]$ guarantee that the split is constant on each $x = a$ block. Soundness constructs Y, Z and observes block-consistency from the B-constraints; completeness chooses $B[a]$ according to which branch contains the block and sets T_α, T_β accordingly.

This completes the inductive proof of the semantic equivalence $\mathcal{A} \models_T \varphi \Leftrightarrow \exists U\,\mathrm{Comp}_0(\varphi; T, U)$.

Logspace-uniformity. Fix φ (as in the theorem). The construction of $\mathrm{Comp}_0(\varphi; T, U)$ proceeds by traversing the syntax tree of φ and, for each constructor, emitting the corresponding clause schemata by iterating over A^r (and over A where quantifier witnesses are required, or over pairs $(t, t') \in A^r \times A^r$ for pairwise dependency conditions). These iterations can be done with $O(\log|A|)$ workspace using lexicographic counters and on-the-fly evaluation of the required coordinate equalities and relation membership checks in $\mathcal{A}$. No clause requires storing more than $O(1)$ tuples simultaneously, and all indices are computable from loop counters, hence the compilation is logspace-uniform.

Fragment preservation. Each syntactic construct contributes constraints of a fixed propositional shape: FO literals produce unit clauses; conjunction adds no new clauses beyond conjunction; disjunction adds clauses $T \to (T_\alpha \vee T_\beta)$ (dual-Horn) and $T_\alpha \to T$, $T_\beta \to T$ (Horn); existential quantification contributes clauses with a single positive head (Horn) plus definitional reachability-style clauses whose polarity is fixed by the chosen encoding; universal quantification yields Horn implications; block disjunction adds Horn implications guarded by selectors. For dependency atoms: dependence/constancy/exclusion yield Horn 2-clauses $\neg p \vee \neg q$; inclusion yields dual-Horn clauses $\neg p \vee (q_1 \vee \cdots \vee q_m)$; anonymity yields Horn implications $\neg p \vee q$; and conditional independence yields Horn 3-clauses $\neg p \vee \neg q \vee r$. Consequently, if φ belongs to a fragment whose compilation rules restrict these patterns to Horn, dual-Horn, 2-CNF, implicative 2-CNF, or 1-EDH, then the resulting matrix lies in the corresponding propositional class. Because the QBF encodings of implication, abduction, and repair only *quantify* over the team-encoding variables (and introduce at most standard definitional or cardinality interfaces), embedding $\exists U\,\mathrm{Comp}_0(\varphi; T, U)$ into those skeletons preserves the matrix fragment guarantees; the quantifier prefix changes, but the clause patterns contributed by φ do not. This establishes the theorem. □

4. Complexity consequences via alternation and matrix fragments

This section explains how the compilation framework yields immediate complexity upper bounds, and how it refines these bounds by tracking two orthogonal sources of difficulty: (i) the *quantifier alternation* forced by the reasoning task (implication versus search), and (ii) the *syntactic matrix fragment* inherited from the compiled semantics of the underlying team logic. The central message is that many known tractability frontiers for model checking reappear here in the *matrix*, while the jump from SAT to QBF is driven primarily by whether the task universally quantifies over teams.

4.1. Implication: why universal quantification forces QBF

Implication is inherently a second-order style question: it asks whether *every* team that satisfies φ also satisfies ψ. Even when model checking for φ and ψ is easy, the universal quantifier over teams typically pushes the problem above NP because the input is no longer a single team but the *space of*

all teams over the relevant vocabulary. Under our compilation, this universal quantification becomes a universal quantifier block over the membership variables $T[t]$.

Recall the compilation correspondence:

$$\mathcal{A} \models_T \theta \quad \Longleftrightarrow \quad \exists U_\theta \, \mathrm{Comp}_0(\theta; T, U_\theta).$$

Using this, implication can be written as the QBF-validity statement

$$\forall T \, \Big((\exists U_\varphi \, \mathrm{Comp}_0(\varphi; T, U_\varphi)) \;\rightarrow\; (\exists U_\psi \, \mathrm{Comp}_0(\psi; T, U_\psi)) \Big).$$

A standard prenexing step converts this to a $\forall\exists$-prefix. The following proposition records the resulting alternation bound and the fact that the reduction can be carried out in logspace.

Proposition 1 (Alternation bound for implication)**.** *Fix φ, ψ in any fragment where Comp_0 has a polynomial-size CNF matrix. The implication problem*

$$\forall X \, (\mathcal{A} \models_X \varphi \Rightarrow \mathcal{A} \models_X \psi),$$

reduces in logspace to validity of a QBF in the class $\forall\exists$ with prefix $\forall T \, \forall U_\varphi \, \exists U_\psi$ and propositional matrix

$$(\neg \mathrm{Comp}_0(\varphi; T, U_\varphi)) \;\vee\; \mathrm{Comp}_0(\psi; T, U_\psi).$$

Explanation of the reduction.

We represent the universally quantified team X by the universally quantified membership family $T[\cdot]$. For each fixed T, the statement $\mathcal{A} \models_T \varphi$ is expressed as $\exists U_\varphi \, \mathrm{Comp}_0(\varphi; T, U_\varphi)$, and similarly for ψ. Thus the implication condition for a fixed T is

$$(\exists U_\varphi \, \mathrm{Comp}_0(\varphi; T, U_\varphi)) \Rightarrow (\exists U_\psi \, \mathrm{Comp}_0(\psi; T, U_\psi)).$$

This is logically equivalent to

$$\forall U_\varphi \, \Big(\mathrm{Comp}_0(\varphi; T, U_\varphi) \Rightarrow \exists U_\psi \, \mathrm{Comp}_0(\psi; T, U_\psi) \Big),$$

and by prenexing we obtain the $\forall\exists$-prefix $\forall T \, \forall U_\varphi \, \exists U_\psi$ with matrix $(\neg \mathrm{Comp}_0(\varphi; T, U_\varphi)) \vee \mathrm{Comp}_0(\psi; T, U_\psi)$. The placement of U_φ in the universal block is essential: if U_φ were existential, one could always *choose* a valuation that falsifies $\mathrm{Comp}_0(\varphi; T, U_\varphi)$, making the disjunction trivially true even when T encodes a team that actually satisfies φ. Finally, if $\mathrm{Comp}_0(\varphi; \cdot)$ and $\mathrm{Comp}_0(\psi; \cdot)$ are in CNF (or are converted to CNF by standard definitional/Tseitin steps), the resulting matrix remains a polynomial-size CNF.

4.2. Bounded abduction and repair: existential search with controlled gadgets

In contrast to implication, bounded abduction and bounded repair are *existential* tasks: we search for a small modification object—an explanation set E to add, or a keep-set K to retain—and then we search for the semantic witnesses U required by the satisfaction derivation of φ. Hence the compilation naturally yields an existential prefix, and the core difficulty resembles SAT unless the underlying logic introduces additional universal conditions.

For bounded abduction, the canonical compiled form is:

$$\exists E \exists S \exists U \, \Big(\mathrm{Card}_{\leq k}(E) \;\wedge\; \bigwedge_{t \in A^r} (S[t] \leftrightarrow (T[t] \vee E[t])) \;\wedge\; \mathrm{Comp}_0(\varphi; S, U) \Big),$$

where T encodes the observed base team, E selects added tuples, and S encodes the augmented team $T \vee E$. The only non-semantic component is the cardinality constraint $\mathrm{Card}_{\leq k}(E)$, which can be implemented using standard polynomial-size CNF encodings.

Downward closure plays a central role in identifying tractable cases. Intuitively, in a downward-closed logic, removing tuples cannot falsify a satisfied formula; such logics often admit Horn-like compilations because satisfaction constraints can be expressed by forbidding local violations rather than requiring global witnesses. This interacts well with abduction and repair because the additional gadgets (cardinality and definitional equivalences) can be treated as a small "interface" around an otherwise Horn/dual-Horn core.

Proposition 2 (SAT-level bound for bounded abduction in downward-closed fragments)**.** *Let $\mathcal{L}$ be a downward-closed team logic fragment in which $\mathrm{Comp}_0(\varphi; T, U)$ is Horn or dual-Horn for every $\varphi \in \mathcal{L}$. Then bounded abduction for fixed φ reduces to satisfiability of a formula of the form*

$$\exists E \exists U\ (\mathrm{Card}_{\leq k}(E) \wedge \mathrm{Comp}_0(\varphi; T \vee E, U)),$$

whose matrix is Horn/dual-Horn plus a standard cardinality gadget. In particular, for constant k the reduction preserves tractable Horn-like structure up to a polynomial-time decidable extension.

Why constant k is special.

When k is a fixed constant, $\mathrm{Card}_{\leq k}(E)$ can be expressed in a way that behaves like a bounded-size disjunction over choices of at most k tuples, or via counters of constant depth, leading to particularly efficient evaluation strategies. Even when one uses generic polynomial-size CNF encodings, the constraint acts as a small perturbation: it restricts the number of $E[t]$ variables that can be set to true, effectively limiting the search space. Thus, in practice, abduction with small k often behaves like "Horn/dual-Horn satisfiability with a small non-Horn backdoor."

Deletion repair.

The same reasoning applies to deletion repair. Introducing keep-variables $K[t]$ and enforcing $K[t] \to T[t]$ yields a repaired team K that is a subteam of the input. The bound $|X \setminus X'| \leq k$ is enforced by a cardinality constraint on indicators $D[t]$ with $D[t] \leftrightarrow (T[t] \wedge \neg K[t])$. The compiled form is

$$\exists K \exists D \exists U\ \Big(\bigwedge_t (K[t] \to T[t])\ \wedge\ \bigwedge_t (D[t] \leftrightarrow (T[t] \wedge \neg K[t]))\ \wedge\ \mathrm{Card}_{\leq k}(D)\ \wedge\ \mathrm{Comp}_0(\varphi; K, U) \Big).$$

When $\mathrm{Comp}_0(\varphi; K, U)$ is Horn or dual-Horn, the semantic core remains in a tractable class, and the only additional hardness is concentrated in the bounded cardinality gadget. This yields a clean complexity explanation: repair difficulty is driven by the combinatorics of selecting deletions, not by the evaluation of the underlying constraint on a fixed team.

Practical message.

The compilation framework therefore draws a sharp line between the tasks. Implication naturally compiles to $\forall\exists$-QBF, even when the model checking cores are easy, because universality over teams is the dominant source of complexity. Bounded abduction and bounded repair, by contrast, compile to SAT-like instances with a small set of additional control constraints (cardinality, definitional equivalences, and, if present, block-consistency selectors). As a result, they can often be handled using SAT, MaxSAT, or incremental SAT workflows, while implication benefits from QBF solving—and in both cases the matrix-fragment information gives a principled way to select solver technology and to anticipate tractable subcases.

5. Illustrative case studies

5.1. Implication between inclusion/anonymity constraints

In a database reading, inclusion and anonymity constraints behave well under union (team) semantics: if two teams satisfy such a constraint, then their union does as well. This union-closure is exactly what makes them compatible with dual-Horn style compilations in the model-checking pipeline: the satisfaction conditions can be expressed as implications where "positive" structure is preserved and negation appears only in a controlled way. Concretely, inclusion-style constraints (e.g., "every value appearing in column set A also appears in column set B") can be seen as universally quantified conditions over tuples in the team, while anonymity constraints (e.g., k-anonymity variants) assert the existence of sufficiently many indistinguishable tuples for each projected pattern. Under standard encodings, these become clauses that are dual-Horn: each clause contains at most one negative literal when written in implication form over indicator variables for tuple membership or equivalence classes.

Implication $\varphi \Rightarrow \psi$ is then the universal question: for every team X, if X satisfies φ it must also satisfy ψ. In the compilation framework, this universal quantification over teams is made explicit by quantifying over propositional variables that represent which tuples are "in" the candidate team (and, when needed, auxiliary variables representing witness structures for constraints). The resulting formula has the shape $\forall T\, \forall U_\varphi\, \exists U_\psi\ \Theta(T, U_\varphi, U_\psi)$, where T encodes the candidate team, U_φ ranges over antecedent-witness variables, and U_ψ provides the existential witnesses needed to establish ψ. The propositional matrix Θ is built as a disjunction

$$\neg\mathrm{Comp}_0(\varphi; T, U_\varphi)\ \vee\ \mathrm{Comp}_0(\psi; T, U_\psi),$$

so that any universally chosen team that makes φ true must also make ψ true, otherwise the matrix is falsified. What matters in this case study is not only that the implication problem lands in $\forall\exists$-QBF, but that the matrix inherits structure: each compiled component remains dual-Horn, and the overall matrix is a "small combination" of dual-Horn fragments. This is valuable in practice because many QBF solvers exploit Horn/dual-Horn backbones, enabling propagation and conflict learning that resembles efficient database inference rather than raw second-order reasoning. Operationally, this gives a principled route from a high-level dependency question ("does this constraint subsume that one?") directly to a solver-ready QBF with structured clauses.

5.2. Abduction as "missing tuples" completion

Abduction in team semantics naturally models data completion: you observe an incomplete team X_0 extracted from noisy logs or partial database snapshots, and you ask whether there exists an extension $X \supseteq X_0$ that satisfies a target constraint φ, subject to a budget limiting how many tuples you may add. This captures, for example, completing missing join results, recovering suppressed rows, or hypothesizing unseen records needed to satisfy a policy constraint. Formally, with a fixed universe of candidate tuples U (derived from active domain, schema bounds, or external candidate generation), abduction asks whether there exists $Y \subseteq U \setminus X_0$ with $|Y| \leq k$ such that $X_0 \cup Y \models \varphi$.

For weak fragments built from inclusion/anonymity constraints, the compiled representation $\mathrm{Comp}_0(\varphi)$ is dual-Horn, which has two practical consequences. First, it allows the abduction problem to be encoded as SAT with a simple "choice layer": introduce a Boolean variable a_t for each candidate tuple $t \in U \setminus X_0$ indicating whether it is added, force all tuples in X_0 to be present, and then impose the compiled dual-Horn constraints over the resulting team indicators. Second, the budget constraint

$\sum_t a_t \leq k$ can be handled in several solver-friendly ways: (i) cardinality constraints via sequential counters or sorting networks for pure SAT, (ii) pseudo-Boolean constraints with PB solvers, or (iii) direct CP-SAT / MILP formulations. Because the underlying constraint part is dual-Horn, unit propagation is strong: once a few additions are chosen, many forced consequences propagate quickly, pruning the search space.

This yields a workflow that is practical rather than purely theoretical. For a fixed k, one solves a satisfiable instance "is there an explanation of size at most k?"; to find a minimal explanation, one can iterate $k = 0, 1, 2, \ldots$ (incremental SAT), or solve a MaxSAT variant where adding tuples has a cost and satisfying φ is hard. The same encoding can also support "explanation diversity" (find multiple distinct completions) by adding blocking clauses after each solution. Importantly, the compilation ensures that the semantics of the constraint φ is respected exactly, so the completion is not an ad-hoc repair but a semantics-preserving explanation within the team framework.

5.3. Repair under block decomposition

Repair problems ask for small modifications of a team so that it satisfies a constraint, often under a cost model (tuple deletions, additions, or edits). When formulas include block disjunction (or other decomposition operators that enforce a partitioning of the team into subteams satisfying different subformulas), naive repairs can easily "cheat" by implicitly mixing tuples across blocks in a way that violates the intended semantics. For instance, if φ expresses that the team can be split into two blocks $X_1 \cup X_2$ where $X_1 \models \varphi_1$ and $X_2 \models \varphi_2$, then a repair that only checks satisfaction of φ_1 and φ_2 on the whole team can accept a solution that does not correspond to any valid partition. This is particularly problematic in settings like policy enforcement or provenance reasoning, where the decomposition carries meaning (e.g., "either this explanation holds on these tuples or that explanation holds on those tuples").

A block-aware encoding prevents this by explicitly representing the decomposition within the propositional variables: for each tuple t, introduce block assignment variables $b_{t,1}, b_{t,2}, \ldots$ indicating which block the tuple belongs to, enforce that each present tuple is assigned to exactly one block, and then compile φ_1 and φ_2 against the induced subteams. Repairs are then searched over both (i) tuple modification variables (delete/add) and (ii) block assignment variables, but the encoding guarantees that any satisfying assignment corresponds to a genuine decomposition satisfying the original semantics. This matters because it turns an error-prone "repair by local checks" approach into a sound global search.

Practically, this block-aware formulation supports several repair modes. If deletions are allowed, one can minimize $\sum_t d_t$ where d_t indicates deletion, subject to satisfiability of the compiled constraints; if additions are allowed, one can similarly control $\sum_t a_t$. If the repair must preserve certain tuples (e.g., legally mandated records), those can be pinned. Moreover, because the decomposition is explicit, one can add domain-specific preferences, such as keeping certain tuples together in the same block, limiting block sizes, or aligning blocks with known categories. The net effect is that the repair process stays faithful to the decomposition semantics by construction, and solver output becomes interpretable: it not only tells you which tuples to change, but also which semantic block each tuple belongs to, yielding a structured explanation of the repair.

6. Discussion

The compilation viewpoint developed in this paper provides a compact explanation for why reasoning in team semantics exhibits a characteristic "two-layer" complexity structure [6, 7]. The first layer is semantic/syntactic: the choice of dependency atoms and connectives determines which propositional fragment (Horn, dual-Horn, 2-CNF, implicative 2-CNF, or special cases such as 1-EDH) arises when compiling satisfaction of a fixed formula on a fixed team. The second layer is task-driven: reasoning problems that quantify over teams, or over team modifications, introduce quantifier alternation over team-encoding variables [13]. A key gain is that these layers separate cleanly: the underlying team logic largely determines the matrix of the compiled formula, while the reasoning task is reflected almost entirely in the QBF prefix. This yields a finer explanation than a single worst-case label and points to solver strategies targeted at the actual source of hardness.

Prototype validation. Beyond proof-level correctness, we implemented (i) a reference interpreter for lax team semantics for the operators used in the paper and (ii) an independent compiler generating the constraints underlying Comp_0. On the smallest nontrivial domains we exhaustively validated the compilation theorem: for $A = \{0, 1\}$, arity $r = 2$ (so $|A^r| = 4$ and 16 teams), and 61 randomly generated formulas of depth up to 3, semantic satisfaction matched satisfiability of $\exists U\, \mathrm{Comp}_0(\varphi; T, U)$ in all 976 team–formula instances. The harness also isolates a necessary guard in existential-quantifier compilation: dropping $W[t, a] \to T_\theta[t]$ allows injecting tuples not generated from the input team (e.g., $X = \{(0, 0)\}$ and $\exists x_1\, \Upsilon x_2$), producing spurious satisfactions in 12 of the 976 instances. Finally, a sanity check on the implication skeleton confirms that antecedent witness variables must be universally quantified: the incorrect prefix $\forall T\, \exists U_\varphi\, \exists U_\psi$ can validate false implications such as $((x_1 = x_1) \vee (x_1 = x_1)) \Rightarrow (x_1 = 0)$ over $A = \{0, 1\}$, whereas the corrected $\forall T\, \forall U_\varphi\, \exists U_\psi$ agrees with the semantic definition.

A central observation is that implication behaves fundamentally differently from bounded abduction and bounded repair. Even when model checking for a fragment reduces to satisfiability in a tractable propositional class, implication introduces universal quantification over the team encoding. Under the membership-variable representation, the input team becomes a universally quantified Boolean vector indexed by tuples of A^r, placing implication naturally at the $\forall\exists$ level once semantic witnesses are accounted for [2, 6]. This matches the intuition that implication is closer to entailment or static analysis (ranging over all admissible datasets) than validation on a single instance, and it parallels classic database containment/entailment phenomena. The compilation framework makes the increase explicit: as soon as the task requires $\forall X$, team variables move under a universal quantifier regardless of how easy evaluation is for a fixed team [7–9].

At the same time, polynomial-hierarchy placement is only part of the story. The compiled implication QBF is not an arbitrary $\forall\exists$ instance: its matrix is assembled modularly from the matrices obtained for φ and ψ. When these matrices lie in Horn, dual-Horn, or 2-CNF, the combined instance can retain a structured backbone even if superficial syntactic transformations move it outside a pure fragment [14]. Practically, this suggests that implication instances from restricted team logics may have small backdoors to tractable classes, benefiting from QBF preprocessing or hybrid SAT–QBF methods. It also motivates identifying fragments where the disjunctive combination

$$(\neg\mathrm{Comp}_0(\varphi; T, U_\varphi)) \;\vee\; \mathrm{Comp}_0(\psi; T, U_\psi),$$

admits normalization that preserves tractability-relevant structure (e.g., bounded incidence width,

bounded clause size, or monotone structure in universally quantified variables). Thus, matrix-fragment tracking can guide both theory and solver-friendly fragment design.

In contrast, bounded abduction and bounded repair typically remain existential search problems. The unknown object is a modification of a given team: an explanation set E (abduction) or a keep-set K / deletion set (repair). After choosing the modification, one witnesses satisfaction of φ on the resulting team via auxiliary variables U, yielding encodings with essentially existential prefixes and a SAT-like shape plus control constraints [14]. The compilation isolates where hardness enters: often not from the semantic core $\mathrm{Comp}_0(\varphi;\cdot)$ (when φ is tractable) but from task-level interface gadgets such as cardinality constraints and definitional equivalences constructing the modified team. This decomposition is practically useful: abduction/repair can often be attacked with SAT-based techniques even when implication for the same fragment requires QBF [15].

Downward closure is particularly instructive for abduction/repair. In many downward-closed fragments, satisfaction is preserved under subteams, enabling Horn-style "no-violation" formulations. Bounded abduction, however, adds tuples, and additions may introduce new violations. The framework clarifies that the main difficulty is searching over constrained augmentations, not checking φ once an augmentation is fixed. When the budget k is small, the cardinality gadget acts as a limited "control layer" around a Horn/dual-Horn core [16], suggesting incremental SAT/MaxSAT workflows where the semantic core is reused and only the budget layer changes. For deletion repair, the alignment with downward closure is stronger: removing tuples cannot create new violations in many fragments, so repair becomes selecting a sufficiently large consistent subteam. The encoding isolates the repair choice in $K[t]$ and bounds deletions by constraining $T[t] \wedge \neg K[t]$. If $\mathrm{Comp}_0(\varphi; K, U)$ is Horn or dual-Horn, verification stays easy and the dominant combinatorial part is tuple selection, naturally mapping to SAT with counters (decision repair), MaxSAT (minimum repair), or SAT-based enumeration for all repairs [1, 7, 10].

The framework also clarifies why block-aware reasoning is more than syntactic ornamentation. Block disjunction enforces structured splitting: assignments with the same block key must be routed together, matching locality/grouping constraints in data practice. Compilation introduces selector variables indexed by domain values and guarded implications

$$T[t] \wedge B[\pi_x(t)] \to T_\alpha[t], \qquad T[t] \wedge \neg B[\pi_x(t)] \to T_\beta[t],$$

which are Horn-style constraints. Choices shift from per-tuple to per-block: this can reduce branching (fewer independent decisions) but can also introduce global coupling across tuples, affecting optimization difficulty. Because the semantic core remains fixed, the compilation enables principled experiments on how block granularity impacts solver performance.

Independence atoms add another dimension. Conditional independence $\vec{y} \perp_{\vec{x}} \vec{z}$ requires witnesses inside the team: for each compatible pair (s, s') there must exist an amalgamated assignment s''. We provided one natural compilation using Horn 3-clauses enforcing closure under amalgamation, but alternative encodings can move complexity between matrix structure and auxiliary witness layers [11, 12]. This suggests refining encodings to preserve a large Horn/low-width backbone while isolating the hard part into a small set of variables (a backdoor), improving preprocessing and learning. By contrast, inclusion and anonymity atoms compile naturally to dual-Horn and Horn respectively, explaining their favorable behavior and suggesting that fragments centered on them may be especially suitable for abduction/repair workflows.

Overall, the separation between prefix-induced hardness and matrix-induced hardness yields a design heuristic for applications. If implication checks are required, the universal quantifier over

teams is unavoidable, so one should keep compiled matrices as structured as possible by restricting to fragments with Horn/dual-Horn/2-CNF compilations and controlling operators (e.g., disjunction and existential quantification) that introduce large auxiliary layers [5, 6]. If the main tasks are completion, abduction, or repair, existential search dominates, and one can combine a tractable semantic core with mature SAT/MaxSAT tooling for the budgeted modification layer.

Finally, several limitations qualify the present complexity discussion. (i) Results are stated for fixed formulas and finite structures; polynomial-size compilation relies on bounded arity and treating φ as constant. If φ is part of the input, formula size must be accounted for explicitly, and naive expansion may cause exponential blow-ups. (ii) Beyond Horn/dual-Horn/2-CNF membership, practical performance depends on structural parameters such as incidence treewidth, variable ordering, and definitional equivalences introduced by CNF conversion. (iii) Bounded abduction/repair are proxies for optimization problems: minimum repair and minimum explanation lead naturally to MaxSAT, QMaxSAT, or optimization-QBF. While the compilation still applies, complexity and solvability depend strongly on the chosen optimization architecture.

7. Conclusion

We developed a modular compilation framework from team-semantics reasoning tasks that quantify over teams (implication, abduction, repair) into QBF, extending the SAT-compilation paradigm beyond model checking. The results position QBF as the natural target for universal or mixed-quantifier questions about teams, and they expose promising low-complexity instances when the compiled matrices retain Horn/dual-Horn/2-CNF backbones. Future work includes sharper lower bounds for specific fragments, systematic treatment of independence atoms via controlled witness layers, and empirical evaluation using modern QBF solvers on database-inspired benchmarks.

References

[1] Durand, A., Kontinen, J., & Väänänen, J. (2024). Modular SAT-based techniques for reasoning tasks in team semantics. *Journal of Computer and System Sciences, 146*, 103575.

[2] Väänänen, J. (2007). *Dependence Logic: A New Approach to Independence Friendly Logic* (Vol. 70). Cambridge University Press.

[3] Abramsky, S., Kontinen, J., Väänänen, J., & Vollmer, H. (Eds.). (2016). *Dependence Logic: Theory and Applications*. Birkhäuser.

[4] Farah, I., Montalbán, A., Zilber, B., Avigad, J., Blass, A. R., Jones, G., ... & Levin, L. A. (2014). Manchester, UK July 12–18, 2012. *The Bulletin of Symbolic Logic, 20*(3), 369-410.

[5] H. Kleine Büning and U. Bubeck, *Theory of Quantified Boolean Formulas*, in A. Biere, M. Heule, H. van Maaren, and T. Walsh (eds.), *Handbook of Satisfiability*, IOS Press, 2009.

[6] Lohmann, P., & Vollmer, H. (2013). Complexity results for modal dependence logic. *Studia Logica, 101*(2), 343-366.

[7] Durand, A., Kontinen, J., & Vollmer, H. (2016). Expressivity and complexity of dependence logic. In *Dependence Logic: Theory and Applications* (pp. 5-32). Cham: Springer International Publishing.

[8] Abiteboul, S., Hull, R., & Vianu, V. (Eds.). (1995). *Foundations of Databases: The Logical Level*. Addison-Wesley Longman Publishing Co., Inc..

[9] Armstrong, W. W. (1974, August). Dependency structures of data base relationships. In *IFIP congress* (Vol. 74, pp. 580-583).

[10] Chen, X., Jia, S., & Xiang, Y. (2020). A review: Knowledge reasoning over knowledge graph. *Expert Systems With Applications, 141*, 112948.

[11] Gutsfeld, J. O., Meier, A., Ohrem, C., & Virtema, J. (2022, August). Temporal team semantics revisited. In *Proceedings of the 37th Annual ACM/IEEE Symposium on Logic in Computer Science* (pp. 1-13).

[12] Biere, A., Heule, M., & van Maaren, H. (Eds.). (2009). *Handbook of Satisfiability* (Vol. 185). IOS press.

[13] Aliseda, A. (2017). The logic of abduction: an introduction. In *Springer Handbook of Model-Based Science* (pp. 219-230). Cham: Springer International Publishing.

[14] Lian, X., Chen, L., & Song, S. (2010, June). Consistent query answers in inconsistent probabilistic databases. In *Proceedings of the 2010 ACM SIGMOD International Conference on Management of data* (pp. 303-314).

[15] Stockmeyer, L. J. (1976). The polynomial-time hierarchy. *Theoretical Computer Science, 3*(1), 1-22.

[16] Borraz-Sánchez, C., Klabjan, D., Pasalic, E., & Aref, M. (2018). SolverBlox: algebraic modeling in datalog. In *Declarative Logic Programming: Theory, Systems, and Applications* (pp. 331-354).

How to cite this article: Fei Yu (2025). Modular QBF Compilations for Implication, Abduction, and Repair in Team Semantics. *Bulletin of Computer and Data Sciences*, 6(4), 1-21. DOI: 10.71448/bcds2564-1

Received: 21/06/2025 **Revised:** 21/09/2025 **Accepted:** 19/11/2025 **Publish:** 30/12/2025