Logical Implication for Full Dependencies

Jef Wijsen

April 28, 2023

Full Dependencies

Full dependencies are closed formulas of the following form:

equality generating (fegd)

$$\forall \vec{x} \left(\overbrace{R_1(\vec{x}_1) \land \cdots \land R_\ell(\vec{x}_\ell)}^{\text{premise}} \rightarrow s = t \right)$$

where each of s, t is either a variable that also occurs in the premise or a constant.

tuple generating (ftgd)

$$\forall \vec{x} \, (R_1(\vec{x}_1) \land \cdots \land R_\ell(\vec{x}_\ell) \rightarrow S(\vec{y}))$$

where each variable that occurs in $\vec{y}$ also occurs in the premise.

Note: The quantifier block $\forall \vec{x}$ will be omitted.

Example

- Functional dependencies are fegds.
  Multivalued and join dependencies are ftgds.
-

P	$SS\#$	$Name$	$Birth$	Nat
	123	Smith	1964	USA
	456	Jones	1970	GB

N	Nat
	USA
	GB
	NL

$$
\begin{array}{rcl}
P(x, y_1, z_1, w_1), P(x, y_2, z_2, w_2) & \rightarrow & y_1 = y_2 \\
P(x, y_1, z_1, w_1), P(x, y_2, z_2, w_2) & \rightarrow & z_1 = z_2 \\
P(x, y_1, z_1, w_1), P(x, y_2, z_2, w_2) & \rightarrow & w_1 = w_2 \\
P(x, y, z, w) & \rightarrow & N(w) \\
P(x, y, z, \text{Europe}) & \rightarrow & 0 = 1
\end{array}
$$

Logical Implication

Definition (Logical implication)

Let Σ be a finite set of full dependencies, and let σ be a full dependency. We say that Σ logically implies σ, denoted $\Sigma \models \sigma$, if every database instance that satisfies all dependencies of Σ also satisfies σ.

Question: Is there an algorithm that takes as input some Σ and σ, and returns "yes" if $\Sigma \models \sigma$, and "no" otherwise?

Explain why the definition of $\models$ cannot be used as an algorithm.

The Chase Algorithm

Question: Does Σ logically imply
some fegd $R_1(\vec{x}_1) \wedge \cdots \wedge R_\ell(\vec{x}_\ell) \rightarrow s = t$
or some ftgd $R_1(\vec{x}_1) \wedge \cdots \wedge R_\ell(\vec{x}_\ell) \rightarrow S(\vec{y})$?

Algorithm (sketch)

1. Start with $\overbrace{\{R_1(\vec{x}_1), \ldots, R_\ell(\vec{x}_\ell)\}}^{\text{canonical database}} \rightarrow$ *right-hand side*.
2. Minimally modify this canonical database in order to satisfy all dependencies in Σ:
   - an fegd of Σ may force you to make two variables equal (by a substitution), or to make a variable equal to a constant (by a valuation). Always make the same changes to the right-hand side (i.e., to $s = t$ or $S(\vec{y})$);
   - an ftgd of Σ may force you to add a fact to the canonical database.
3. Return *"yes, because there is no counterexample"* as soon as you are forced to make two distinct constants equal (denoted by $\lightning$). If you do not end with $\lightning$ but your final dependency is trivial, also return *"yes, because there is no counterexample"*; otherwise return *"no, because I found a counterexample"*.

Example

$$\begin{aligned}\sigma_1 &: \quad Knows(x,y), Knows(y,z) \rightarrow A(x,z) \\ \sigma_2 &: \quad Knows(x,u), Knows(v,z) \rightarrow A(x,z)\end{aligned}$$

Does $\{\sigma_2\}$ logically imply σ_1?
Here is a chase of σ_1 by $\{\sigma_2\}$:

$$\begin{aligned}\sigma_1 &: \quad Knows(x,y), Knows(y,z) \rightarrow A(x,z) \\ \text{Apply } \sigma_2 &: \quad Knows(x,y), Knows(y,z), A(x,z) \rightarrow A(x,z) \\ \text{Apply } \sigma_2 &: \quad Knows(x,y), Knows(y,z), A(x,z), A(y,y) \rightarrow A(x,z)\end{aligned}$$

Since the last ftgd is trivial, return "yes, it is the case that $\{\sigma_2\} \models \sigma_1$."

Example

$$
\begin{aligned}
\sigma_1 &: \quad \mathit{Knows}(x,y), \mathit{Knows}(y,z) \rightarrow A(x,z) \\
\sigma_2 &: \quad \mathit{Knows}(x,u), \mathit{Knows}(v,z) \rightarrow A(x,z)
\end{aligned}
$$

Does $\{\sigma_1\}$ logically imply σ_2?
Here is a chase of σ_2 by $\{\sigma_1\}$:

$$\sigma_2 : \quad \mathit{Knows}(x,u), \mathit{Knows}(v,z) \rightarrow A(x,z)$$

Since σ_1 is not applicable, the chase immediately terminates.
The canonical database $\{\mathit{Knows}(x,u), \mathit{Knows}(v,z)\}$ satisfies $\{\sigma_1\}$ and falsifies σ_2, hence $\{\sigma_1\} \not\models \sigma_2$.

When viewed as conjunctive queries: $\sigma_1 \sqsubseteq \sigma_2$ but $\sigma_2 \not\sqsubseteq \sigma_1$.

Example

Does $\{\bowtie [AC, ABD], B \to C\}$ logically imply $A \to C$? Let

$$
\begin{aligned}
\sigma_1 &: R(x, y', z, w'), R(x, y, z', w) \to R(x, y, z, w) \\
\sigma_2 &: R(x_1, y, z_1, w_1), R(x_2, y, z_2, w_2) \to z_1 = z_2 \\
\sigma_3 &: R(x, y_1, z_1, w_1), R(x, y_2, z_2, w_2) \to z_1 = z_2
\end{aligned}
$$

Obviously, $\sigma_1 \equiv \bowtie [AC, ABD]$, $\sigma_2 \equiv B \to C$, and $\sigma_3 \equiv A \to C$.

Here is a chase of σ_3 by $\{\sigma_1, \sigma_2\}$:

$$
\begin{aligned}
\sigma_3 &: R(x, y_1, z_1, w_1), R(x, y_2, z_2, w_2) \to z_1 = z_2 \\
\text{Apply } \sigma_1 &: R(x, y_1, z_1, w_1), R(x, y_2, z_2, w_2), R(x, y_2, z_1, w_2) \to z_1 = z_2 \\
\text{Apply } \sigma_2 &: R(x, y_1, z_1, w_1), R(x, y_2, z_1, w_2) \to z_1 = z_1
\end{aligned}
$$

Since the last fegd is trivial, return "yes, it is the case that $\{\sigma_1, \sigma_2\} \models \sigma_3$."

Example

Does $\{R(x) \to x = a, R(x) \to x = b\}$ logically imply $R(v) \to S(v)$?

A chase of $R(v) \to S(v)$ by $\{R(x) \to x = a,\ R(x) \to x = b\}$:

$$\begin{aligned} \text{Initial fegd} &\quad : \quad R(v) \to S(v) \\ \text{Apply } R(x) \to x = a &\quad : \quad R(a) \to S(a) \\ \text{Apply } R(x) \to x = b &\quad : \quad a = b \ \lightning \end{aligned}$$

Since we are forced to make a and b equal, return "yes, it is the case that $\{R(x) \to x = a, R(x) \to x = b\} \models R(v) \to S(v)$."

Example

Let

$$\begin{aligned}
\sigma_1 &: R(x,y) \to R(y,x) \\
\sigma_2 &: R(x,y), S(y,z), R(z,u), S(u,x) \to y = u \\
\sigma_3 &: R(x,y), S(y,z), R(z,u), S(u,x) \to S(x,u)
\end{aligned}$$

Does $\{\sigma_1, \sigma_2\}$ logically imply σ_3?
A chase of σ_3 by $\{\sigma_1, \sigma_2\}$:

$$\begin{aligned}
\sigma_3 &: R(x,y), S(y,z), R(z,u), S(u,x) \to S(x,u) \\
\text{Apply } \sigma_2 &: R(x,u), S(u,z), R(z,u), S(u,x) \to S(x,u) \\
\text{Apply } \sigma_1 &: R(x,u), R(u,x), S(u,z), R(z,u), S(u,x) \to S(x,u) \\
\text{Apply } \sigma_1 &: R(x,u), R(u,x), S(u,z), R(z,u), R(u,z), S(u,x) \to S(x,u)
\end{aligned}$$

The canonical database $\{R(x,u),\ R(u,x),\ S(u,z),\ R(z,u),\ R(u,z),\ S(u,x)\}$ satisfies $\{\sigma_1, \sigma_2\}$ and falsifies σ_3, hence $\{\sigma_1, \sigma_2\} \not\models \sigma_3$.

Chase finds a most general counterexample (if it exists)

Assume $R[ABCDE]$. We have $\{A \to B, B \to C\} \not\models A \to D$, as witnessed by

$$I = \begin{array}{c|ccccc} R & A & B & C & D & E \\ \hline & a & b & c & d_1 & e \\ & a & b & c & d_2 & e \end{array}$$

Start at $A \to D$: $R(u, v_1, w_1, x_1, y_1), R(u, v_2, w_2, x_2, y_2) \to x_1 = x_2$

There is a valuation ν mapping $R(u, v_1, w_1, x_1, y_1)$ to $R(a, b, c, d_1, e)$, and $R(u, v_2, w_2, x_2, y_2)$ to $R(a, b, c, d_2, e)$.

Apply $A \to B$: We apply $\{v_2 \mapsto v_1\}$ giving
$R(u, v_1, w_1, x_1, y_1), R(u, v_1, w_2, x_2, y_2) \to x_1 = x_2$

Apply $B \to C$: We apply $\{w_2 \mapsto w_1\}$ giving
$\underbrace{R(u, v_1, w_1, x_1, y_1), R(u, v_1, w_1, x_2, y_2)}_{J} \to x_1 = x_2$

The substitution $\mu := \{v_2 \mapsto v_1, w_2 \mapsto w_1\}$ maps the "body" of $A \to D$ to J, while $\mu(x_1) = x_1$ and $\mu(x_2) = x_2$. Thus, J is a "counterexample"!

$\nu = \{u \mapsto a, v_1 \mapsto b, w_1 \mapsto c, x_1 \mapsto d_1, y_1 \mapsto e, x_2 \mapsto d_2, y_2 \mapsto e, \dots\}$ is a homomorphism from J to I. Note that y_1 and y_2 are both mapped to e.

Proof: If $\Sigma \not\models$ fegd, the chase ends with a 'counterexample'

Assume $\Sigma \not\models L_0 \to s_0 = t_0$. There exist *(i)* a database instance I s.t. $I \models \Sigma$, and *(ii)* a valuation ν s.t. $\nu(L_0) \subseteq I$ and $\nu(s_0) \neq \nu(t_0)$.

One can show:

Assume the chase sequence is:

$$\begin{array}{ccc} L_0 & \to & s_0 = t_0 \\ L_1 & \to & s_1 = t_1 \\ \vdots & & \vdots \\ L_n & \to & s_n = t_n \end{array}$$

$$\begin{array}{c|ccc} \nu(L_0) \subseteq I & \nu(s_0) & \neq & \nu(t_0) \\ & \| & & \| \\ \nu(L_1) \subseteq I & \nu(s_1) & & \nu(t_1) \\ & \| & & \| \\ \vdots & \vdots & & \vdots \\ & \| & & \| \\ \nu(L_n) \subseteq I & \nu(s_n) & & \nu(t_n) \end{array}$$

Thus $s_n \neq t_n$.

- Informally, each L_{i+1} is obtained from L_i by eliminating a variable (apply fegd), or by adding an atom (apply ftgd).
- There is a substitution μ s.t. $\mu(L_0) \subseteq L_n$, $\mu(s_0) = s_n$, and $\mu(t_0) = t_n$. Thus, $L_n \not\models L_0 \to s_0 = t_0$. Informally, μ combines all chase steps.
- The canonical database L_n will satisfy Σ.

Argumentation why $\nu(L_{i+1}) \subseteq I$

Assume we already established $\nu(L_i) \subseteq I$. Assume the chase step:

$$\boxed{\begin{array}{rcl} \cdots & : & L_i \to s_i = t_i \\ \text{Apply } L \to x = c & : & L_{i+1} \to s_{i+1} = t_{i+1} \end{array}}$$

Then, there was a substitution θ for the variables in L such that

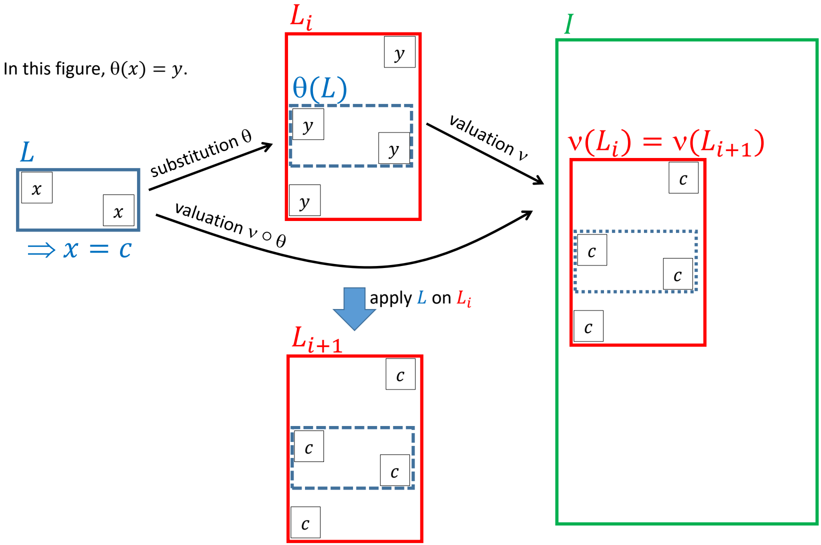
- $\theta(L) \subseteq L_i$, and
- $L_{i+1} = (L_i)_{\theta(x) \to c}$ where $\theta(x)$ is a variable (otherwise the chase would have terminated with $\lightning$).

From $\theta(L) \subseteq L_i$ and $\nu(L_i) \subseteq I$, it follows $\nu \circ \theta(L) \subseteq I$.
Since $I \models L \to x = c$, we have $\nu \circ \theta(x) = c$.
Therefore, $\nu\left((L_i)_{\theta(x) \to c}\right) = \nu(L_i)$ $\begin{bmatrix}\text{and, by analogous reasoning,} \\ \nu(s_{i+1}) = \nu(s_i) \text{ and } \nu(t_{i+1}) = \nu(t_i)\end{bmatrix}$.
Thus, $\nu(L_{i+1}) = \nu(L_i) \subseteq I$.

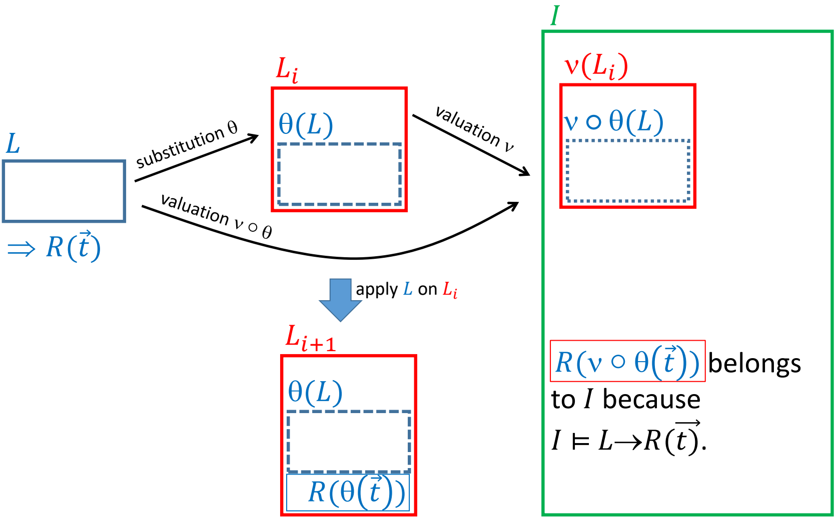
Exercise: Extend the previous reasoning for an application of $L \to x = y$ or $L \to S(\vec{y})$.

Application of fegd $L \to (x = c)$ on L_i

Recall: $I \models L \to (x = c)$.



Application of ftgd $L \to R(\vec{t})$ on L_i



Proof: If $\Sigma \not\models$ ftgd, the chase ends with a 'counterexample'

Assume $\Sigma \not\models L_0 \to S(\vec{y}_0)$. There exist *(i)* a database instance I s.t. $I \models \Sigma$, and *(ii)* a valuation ν s.t. $\nu(L_0) \subseteq I$ and $S(\nu(\vec{y}_0)) \notin I$.

Assume the chase sequence is:

$$\begin{array}{ccc} L_0 & \to & S(\vec{y}_0) \\ L_1 & \to & S(\vec{y}_1) \\ \vdots & & \vdots \\ L_n & \to & S(\vec{y}_n) \end{array}$$

One can show:

$$\begin{array}{c|cc} \nu(L_0) \subseteq I & S(\nu(\vec{y}_0)) & \notin I \\ & \| & \\ \nu(L_1) \subseteq I & S(\nu(\vec{y}_1)) & \\ & \| & \\ \vdots & \vdots & \\ & \| & \\ \nu(L_n) \subseteq I & S(\nu(\vec{y}_n)) & \end{array}$$

Thus $S(\vec{y}_n) \notin L_n$.[†]

- There is a substitution μ s.t. $\mu(L_0) \subseteq L_n$ and $S(\mu(\vec{y}_0)) = S(\vec{y}_n)$. Thus, $L_n \not\models L_0 \to S(\vec{y}_0)$. Informally, μ combines all chase steps.
- The canonical database L_n will satisfy Σ.

[†] $S(\vec{y}_n) \in L_n$ would imply $S(\nu(\vec{y}_n)) \in \nu(L_n) \subseteq I$, a contradiction.

Discussion I

- The database L_n constructed by our proof is thus a counterexample for $\Sigma \models \sigma$, i.e., $L_n \models \Sigma$ and $L_n \not\models \sigma$ (when distinct variables in L_n are treated as distinct constants).
- The proof shows that L_n is homomorphic to I (i.e., there exists a valuation ν that maps every tuple of L_n to a tuple of I).
- Notice that the proof goes through for every database I such that $I \models \Sigma$ and $I \not\models \sigma$.
- Thus, our counterexample is very special: it is homomorphic to every database I that satisfies Σ and falsifies σ. Informally, the counterexample constructed in the proof is the most general possible.

Discussion II

At some point in the chase, more than one full dependency may be applicable. If this happens, we choose—in a non-deterministic way—an applicable full dependency and apply it. Does the outcome of the chase depend on the order in which full dependencies are applied?

- Assume two distinct chase sequences such that one chase sequence terminates with a counterexample L_n for $\Sigma \models \sigma$, thus $\Sigma \not\models \sigma$.
- Then, by what we proved before, the other chase sequence will necessarily also find some counterexample, say L'.
- Then, L_n will be homomorphic to L', and L' will be homomorphic to L_n.

A Note on Non-Full Tuple Generating Dependencies

$$\begin{array}{lcl} \sigma_1 & : & R(u,v) \rightarrow R(v,u) \\ \sigma_2 & : & R(x,y) \rightarrow \exists z\,(S(y,z)) \\ \sigma_3 & : & S(x,y) \rightarrow \exists z\,(R(y,z)) \end{array}$$

Does $\{\sigma_2, \sigma_3\}$ logically imply σ_1?

The chase of $\{R(u,v)\}$ with σ_2 and σ_3 yields

$$\{R(u,v), S(v,z_1), R(z_1,z_2), S(z_2,z_3), R(z_3,z_4), S(z_4,z_5), \ldots\}.$$

But a counterexample for $\{\sigma_2, \sigma_3\} \models \sigma_1$ must be finite.

Optimization of Conjunctive Queries

Consider the (minimal) conjunctive query

$$q \quad : \quad \mathit{Answer}(u, v, w) \leftarrow R(u, v), R(u, w), T(v, w).$$

Assume that this query is executed on databases satisfying the following fegd:

$$\sigma \quad : \quad R(x, y) \wedge R(x, z) \rightarrow y = z.$$

The following query is obtained by a chase of q with $\{\sigma\}$:

$$q' \quad : \quad \mathit{Answer}(u, v, v) \leftarrow R(u, v), T(v, v)$$

Explain: For each database I satisfying σ, we have $q(I) = q'(I)$.

(See the course notes for a more involved example.)

Exercise

Show that $\{A \to C, B \to C, C \to D, DE \to C, CE \to A\}$ logically implies $\bowtie [AD, AB, BE, CDE, AE]$, where the set of attributes is $ABCDE$.

(See the course notes for more exercises.)

Epilogue for Students of *Logique mathématique I*

Most theorems (compactness theorem, completeness theorem, Löwenheim-Skolem theorem) from classical model theory fail in the finite case. See also [Lib04].

Theorem (Compactness)

A theory T is consistent iff every finite subset of T is consistent.

Theorem

Compactness fails over finite models: there is a theory T such that

1. *T has no finite models, and*
2. *every finite subset of T has a finite model.*

Proof.

Let R be a unary relation name. Let $T = \{|R| \geq 0, |R| \geq 1, |R| \geq 2, \ldots\}$, where $|R| \geq n$ is the sentence

$$\exists x_1 \cdots \exists x_n \left(\bigwedge_{1 \leq i \leq n} R(x_i) \wedge \bigwedge_{1 \leq i < j \leq n} x_i \neq x_j \right).$$

$\square$

A Glimpse of *Knowledge Representation and Reasoning*

A subfield of Artificial Intelligence.

Beyond Datalog Can the vertices of a graph (V, E) be colored with three colors such that no two adjacent vertices have the same color?

$$C(x, \text{blue}) \vee C(x, \text{red}) \vee C(x, \text{green}) \leftarrow V(x)$$

$$\text{FALSE} \leftarrow E(x, y), x \neq y, C(x, z), C(y, z)$$

Description Logics Sublanguages of first-order logic with "good" properties (e.g., decidability of logical implication), used in practical applications like the Semantic Web.

More to come. . .

References

Leonid Libkin.
Elements of Finite Model Theory.
Texts in Theoretical Computer Science. An EATCS Series. Springer, 2004.