Application Patterns of Projection/Forgetting

Christoph Wernhard
Technische Universität Dresden

Interpolation: From Proofs to Applications (iPRA 2014)
Vienna, 17 July 2014
Introduction

We assume a classical logic setting where projection and forgetting are available as second-order operators that can be nested.

It allows to define concepts such as:

- Literal projection, literal forgetting
- Globally strongest necessary and weakest sufficient condition
- Definability and definientia

A variety of applications can be rendered with these:

- **View-based query processing**
  - Query rewriting
  - Characterizing definientia in formula classes
- **Knowledge base modularization**
  - Conservative theory extension
- **“Non-standard inferences”**
  - “Formula matching”
- **Non-monotonic reasoning and logic programming**
  - Stable and partial stable model semantics
  - Abduction w.r.t. these semantics
Classical Logic + Second-Order Operators

- We start with an underlying classical logic, e.g., first-order or propositional

- It is extended by second-order operators, e.g., predicate quantification or Boolean quantification

\[ \exists q (p \rightarrow q) \land (q \rightarrow r) \]

- The associated computation is second-order operator elimination: computing an equivalent formula without second-order operators

\[ \exists q (p \rightarrow q) \land (q \rightarrow r) \equiv p \rightarrow r. \]
Forgetting, Projection, Uniform Interpolants

- **Further second-order operators** can be defined in terms of predicate quantification

- An operator for **forgetting** can be seen as syntax for iterated existential predicate quantification:
  \[ \text{forgetAboutPredicates}_{\{p,q\}}(F) \equiv \exists p \exists q F \]

- Elimination of forgetAboutPredicates is often called **computation of forgetting**

- Forgetting about all predicates except those explicitly specified is often called **projection** [Darwiche 01]
  \[ \text{projectOntoPredicates}_{\{p,q\}}(F) \equiv \text{forgetAboutPredicates}_{\text{ALLPREDICATES}\setminus\{p,q\}}(F) \]

- Elimination of projectOntoPredicates is often called **computation of a uniform interpolant**

- Here we handle projection and forgetting **symmetrically as second-order operators**
Scopes as Parameters of Second-Order Operators

- The introduced second-order operators have a set of predicates as parameter. We generalize this to a set of ground literals, called scope.

- A scope can express different effects on positive and negative predicate occurrences.

Our basic second-order operators are now literal projection and literal forgetting:

Let \( F = (p \rightarrow q) \land (q \rightarrow r) \)

\[
\text{forget}_{\neg q}(F) \equiv \text{project}_{\{p,q,r,\neg p,\neg r\}}(F) \equiv (p \rightarrow q) \land (p \rightarrow r)
\]

[Lang* 03, W 08]

An interpretation is a set of ground literals, containing each ground atom either positively or negatively.

\( I \models \text{project}_S(F) \) iff def There exists a \( J \) s.t. \( J \models F \) and \( J \cap S \subseteq I \).

\( \text{forget}_S(F) \) def \( \text{project}_{\text{ALLGROUNDLITERALS}\setminus S}(F) \).
Notation for “in Scope”

- That $F$ is “in scope” $S$ is written as
  
  $$F \in S$$

Let $F = p \lor \neg q \lor (r \land \neg r)$

- $F \in \{p, \neg q\}$
- $F \in \{p, q, r, s, \neg p, \neg q, \neg r, \neg s\}$
- $F \notin \{p\}$

$F \in S$ iff $def$ $F \equiv \text{project}_S(F)$. 
Globally Strongest Necessary and Weakest Sufficient Condition

- The **globally strongest necessary condition** of $G$ on $S$ within $F$ is the strongest $X \subseteq S$ s.th. $(F \land G) \models X$
  
  It can be expressed by a second-order operator
  
  $\text{gsnc}_{\{p\}}((q \rightarrow p), q) \equiv p$

- The **globally weakest sufficient condition** of $G$ on $S$ within $F$ is the weakest $X \subseteq S$ s.th. $(F \land X) \models G$
  
  It can be expressed by a second-order operator
  
  $\text{gwsc}_{\{p\}}((p \rightarrow q), q) \equiv p$

- The analog concepts in [Lin 01] are not unique modulo equivalence. See also [Doherty* 01, W 12]

Let $\overline{S}$ denote the set of the complements of the members of scope $S$.

$\text{gsnc}_S(F, G) \overset{\text{def}}{=} \text{project}_S(F \land G)$.

$\text{gwsc}_S(F, G) \overset{\text{def}}{=} \neg\text{project}_{\overline{S}}(F \land \neg G)$.
Definition, Definability

• A **definition of $G$ in terms of $S$ within $F$** is a formula $(G \leftrightarrow X)$ such that
  1. $X \subseteq S$, and
  2. $F \models G \leftrightarrow X$

$G$ is the **definiendum**, $X$ is the **definiens**

Note: If $F$ is a sentence, then $F \models G(x) \leftrightarrow X(x)$ iff $F \models \forall x (G(x) \leftrightarrow X(x))$

Let $F = (p \leftrightarrow q \land r) \land (q \rightarrow r)$

- $(p \leftrightarrow q \land r)$ is a definition of $p$ in terms of $\{q, r\}$ within $F$
- $(p \leftrightarrow q)$ is a definition of $p$ in terms of $\{q, r\}$ within $F$

• Existence of a definition is called **definability**

- $p$ is definable in terms of $\{q, r\}$ within $F$
- $p$ is definable in terms of $\{q\}$ within $F$
- $p$ is not definable in terms of $\{r\}$ within $F$

• This is a **semantic** characterization, aka implicit definability
Definition, Definability in Terms of Second-Order Operators

- **Definientia** are exactly those formulas in the scope that are **between the GSNC and the GWSC**

Let $F = (p \leftrightarrow q \land r) \land (q \rightarrow r)$

\[
\text{gsnc}_{\{q,r\}}(F, p) \equiv \text{project}_{\{q,r\}}(F \land p) \equiv q \land r
\]

\[
\text{gwsc}_{\{q,r\}}(F, p) \equiv \neg \text{project}_{\{\neg q, \neg r\}}(F \land \neg p) \equiv q
\]

- **Definability** holds iff the GSNC entails the GWSC

\[
\text{gsnc}_{\{q,r\}}(F, p) \equiv q \land r \models q \equiv \text{gwsc}_{\{q,r\}}(F, p)
\]

\[
\text{gsnc}_{\{q\}}(F, p) \equiv q \models q \equiv \text{gwsc}_{\{q\}}(F, p)
\]

\[
\text{gsnc}_{\{r\}}(F, p) \equiv r \notmodels \bot \equiv \text{gwsc}_{\{r\}}(F, p)
\]

- In case of definability, the GSNC and GWSC provide the strongest and weakest definientia

\[
\text{ISDEFINITION}(X, G, S, F) \iff \text{def} X \subseteq S \text{ and } \text{gsnc}_S(F, G) \models X \models \text{gwsc}_S(F, G).
\]

\[
\text{ISDEFINABLE}(G, S, F) \iff \text{def} \text{gsnc}_S(F, G) \models \text{gwsc}_S(F, G).
\]
View-Based Query Rewriting – Exact Views

[Halevy 01, Calvanese* 07, Marx 07, Nash* 10, Bárány* 13, W 14a]

• Given: \( D \) “database scope” \( \{a, \neg a\} \)
  \( U \) “view scope” \( \{p, \neg p, q, \neg q\} \)
  \( V \in D \cup U \) “view specification” \((p \leftrightarrow a) \land (q \leftrightarrow a)\)
  \( Q \in D \) “query” \( a \)

• The “view extension” of \( V \) wrt. “database” \( DB \in D \) is project\(_U\)(\( DB \land V \))
  \[\text{project}_U(a \land V) \equiv p \land q\]
  \[\text{project}_U(\neg a \land V) \equiv \neg p \land \neg q\]

• “Queries to view extensions can be evaluated particularly well”
  The objective is to find an “exact rewriting” \( R \in U \) s.t. for all \( DB \in D \):
  \[\text{project}_U(DB \land V) \models R \text{ iff } DB \models Q\]

• Assume that all \( R \in U \) are uniquely definable in terms of \( D \) within \( V \)
  \[\text{gsnc}_D(V, p) \equiv a \equiv \text{gwsc}_D(V, p)\]

• Then \( R \) is an exact rewriting iff \( R \) is a definiens of \( Q \) i.t.o. \( U \) within \( V \)
  \[\text{gsnc}_U(V, Q) \equiv (p \land q) \models p \models (p \lor q) \equiv \text{gwsc}_U(V, Q)\]
View-Based Query Rewriting – “Split Rewriting”

[W 14a], related to [Borgida* 10, Franconi* 13]

• Given: \(D\) “database scope”
  \(U\) “view scope”
  \(V \subseteq D \cup U\) “view specification”
  \(Q \subseteq D \cup U\) “query”

• The idea is to rewrite a \(Q \subseteq D \cup U\) to a \(R \subseteq D\) that can be evaluated by the “database system”

• The objective is to find a “split rewriting” \(R \subseteq D\) s.t. for all \(DB \subseteq D\):
  \[DB \models R \text{ iff } DB \land V \models Q\]

• \(R\) is a split rewriting iff \(R \equiv \text{gwsc}_{D}(V,Q)\)
View-Based Query Rewriting – Further Issues

- Investigation of "determinacy" w.r.t. formula classes
  [Segoufin and Vianu 05, Marx 07, Nash* 10, Bárány* 13]

  For $Q$, $V$ in particular formula classes:
  - is the existence of an exact rewriting (definability) decidable?
  - what formula class contains all exact rewritings?
Definientia in Formula Classes

[W 14b]

• So far, we considered definientia in terms of a vocabulary

  **Question:** Can we apply second-order operators also to characterize definientia in efficiently processable formula classes?

• Yes, for the class of formulas that are equivalent to a conjunction of atoms

• This class excludes disjunction and negation and can thus be used to encode other syntactic conditions on the meta level

  e.g., a Krom formula as a conjunction of atoms like \(\text{clause}(p, \neg q)\)

\[I \models \text{project}_S(F) \iff \text{def} \quad \text{There exists a } J \text{ s.t. } J \models F \text{ and } J \cap S \subseteq I.\]
\[I \models \text{diff}_S(F) \iff \text{def} \quad \text{There exists a } J \text{ s.t. } J \models F \text{ and } J \cap S \not\subseteq I.\]
\[\text{glb}(F) \quad \text{def} \quad \text{circ}_\text{NEG}(\neg \text{diff}_\text{NEG}(F)).\]
\[\text{fhub}(F) \quad \text{def} \quad \text{project}_\text{POS}(\text{glb}(F)) \land \text{project}_\text{NEG}(F).\]

**ISCA-Definable** \((G, S, F)\) iff \(\text{glb}(\text{gsnc}_{S \cap \text{POS}}(F, G)) \models \text{gwsc}_{S \cap \text{POS}}(F, G)\).

If **ISCA-Definable** \((G, S, F)\), then

**ISCA-Definiens** (\text{fhub}(\text{gsnc}_{S \cap \text{POS}}(F, G)), G, S, F).
Adding $G$ does not "damage my ontology" $F$

iff "All knowledge about the vocabulary of $F$ that is expressed by $(F \land G)$ is expressed by $F$ alone"

iff $(F \land G)$ is a **conservative extension** of $F$ [W 14a]

iff $G$ is **conservative** within $F$ [Cuenca Grau* 08]

iff $G$ imports $F$ in a **safe** way

iff $F \models \text{project}_{\text{vocab}(F)}(F \land G)$

iff $F \equiv \text{project}_{\text{vocab}(F)}(F \land G)$
“Formula Matching”

- **Concept matching modulo equivalence** is a non-standard inference in description logics [Borgida and McGuinness 96, Baader* 99],

- Here for arbitrary formulas but with single-variable patterns

Given:  
- \( F \): Background formula  
- \( G \): Formula  
- \( H \): Pattern: formula with special atom \( x \)

\[ (p \land q) \lor x \]

- Objective: Find a “matching formula” \( X \) such that

\[ F \models G \leftrightarrow H[x \mapsto X] \]

\[ (p \leftrightarrow q) \leftrightarrow ((p \land q) \lor x) \]

\[ (p \leftrightarrow q) \leftrightarrow ((p \land q) \lor (\neg p \land \neg q)) \]

- **There are two second-order formulas** \( M_1 \) and \( M_2 \) such that solutions are exactly the \( X \) s.th. \( M_1 \models X \models M_2 \)

Basic characterization of \( X \):  
\[ \models \forall x F \land (x \leftrightarrow X) \rightarrow (G \leftrightarrow H) \]

This is equivalent to:  
\[ \exists x F \land \neg x \land \neg(G \leftrightarrow H) \models X \]  
and  
\[ X \models \forall x F \land x \rightarrow (G \leftrightarrow H) \]
Stable Model Semantics for Logic Programming

Let $F = p \land (q \leftarrow p \land \neg r)$

It has three models: $\{p, q, r\}$, $\{p, q, \neg r\}$, $\{p, \neg q, r\}$

Considered as logic program it has a single **stable model**: $\{p, q\}$

- **Logic programs can be represented by classical formulas, where second-order operators associate logic programming semantics** [W 10]

  $$\text{stable}(p \land (q \leftarrow p \land \neg r^1)) \equiv (p \land q \land \neg r)$$

  A “replica” of the vocabulary, identified by the $1$ superscript, is used for predicate occurrences under negation as failure

- **$\text{stable}(F) \overset{\text{def}}{=} \text{rename}_{1\mapsto 0}(\text{circ}_{(0\cap \text{POS})\cup 1}(F))$**

  1. minimize undecorated predicates, while keeping $1$ predicates fixed
  2. rename the $1$ predicates to their undecorated correspondents

- The stable operator renders the characterization of the stable model semantics in terms of circumscription from [Lin 91]

- By combination with an encoding from [Janhunen* 06], a similar operator can render the $3$-valued **partial stable model semantics**
Abduction with the Stable Model Semantics

[Kakas* 98, Lin and You 02, W 13a]

- Given: \( F \) **background**
  \[(wet \leftarrow shower) \quad \land \quad (wet \leftarrow rain \land \neg umbrella) \quad \land \quad (umbrella \leftarrow forecastRain)\]

- \( G \) **observation**
  \( wet \)

- \( S \) **abducibles**
  \( \{shower, rain, forecastRain, \neg shower, \neg rain, \neg forecastRain\} \)

- In classical logic, an **explanation** is an \( X \in S \) s.th. \((F \land X) \models G\)
  
The weakest explanation is \( gwsc_S(F, G) \)

- For the **stable model semantics**, a “factual” **explanation** is a conjunction of literals \( X \in S \) s.th.
  \( stable_S(F \land X) \models G \)
  
  stable\(_S\) effects that atoms occurring in \( S \) are subjected to the open-world assumption (passed as “fixed” to the circumscription)

  The minimal factual explanations for the example are
  \( shower \) and \( (rains \land \neg forecastRain) \)
Abduction with the Stable Model Semantics (2)

For the stable model semantics, a “factual” explanation is a conjunction of literals $X \subseteq S$ s.th.

$$\text{stable}_S(F \land X) \models G$$

- The **minimal factual explanations** are the **prime implicants** of

$$\text{gwsc}_{S \cap 0}(\text{stable}_S(F), G)$$

- $S \cap 0$ specifies the undecorated literals in $S$
- The underlying justification is that for $H \subseteq S \cup \overline{S}$ it holds that

$$\text{stable}_S(F \land H) \equiv \text{stable}_S(F) \land H$$

$$\text{gwsc}_{S \cap 0}(\text{stable}_S(F), G) \equiv \neg \text{project}_{S \cap 0}(\text{stable}_S(F) \land \neg G)$$
Abduction with 3-Valued Logic Programming Semantics

[W 13a]

- Abduction can be analogously characterized with the **GWSC** for
  - the **well founded semantics**
  - the **partial stable model semantics**

- For the partial stable model semantics, this seems so far the only thorough formalization of abduction

- Unlike the well-founded semantics, the partial stable model semantics allows to obtain **explanations for the undefinedness of observations**

Background: The barber shaves all males who do not shave themselves

The barber shaves the barber if the barber has been sentenced to shave himself

Observation: “The barber shaves the barber” is undefined

Explanation: The barber is male and has not been sentenced to shave himself
Conclusion – Towards Practice

• ToyElim [W 13b] is a Prolog-based **prototype system** which supports to define second-order operators as outlined and is useful for small experiments.

• Relevant general processing techniques include:
  • **second-order quantifier elimination methods** based on first-order logic [Gabbay and Ohlbach 92, Doherty* 97]
  • recent advances in **uniform interpolation for description logics** [Ghilardi* 06, Konev* 09, Koopmann and Schmidt 13]
  • progress in **SAT pre- and inprocessing** [Eén and Biere 05, Heule* 10, Manthey* 13]

• General agenda: Investigate processing of the particular **formula patterns** in which combinations of second-order operators are used in applications.
  Consider these patterns also for **restricted argument formulas**.
Conclusion – Classical Logic + Second-Order Operators

- Provides an **integrating view on a variety of applications** in areas such as
  - view-based query processing
  - knowledge base modularization
  - many “non-standard” inferences
  - non-monotonic reasoning and logic programming
  - abductive reasoning

- **Operators can be nested and combined**

- **New operators can be defined in terms of other ones**

- **Operators let instructive relationships become evident**

- **Operators seems useful for mechanization**

- **Second-order operators shift techniques from a theoretical background to a mechanizable and user accessible formalization**
References
Computing the least common subsumer and the most specific concept in the
presence of cyclic $ALN$-concept descriptions.
In KI-98, volume 1504 of LNCS, pages 129–140. Springer.

Matching in description logics.

Rewriting guarded negation queries.
In Mathematical Foundations of Computer Science 2013, volume 8087 of LNCS,
pages 98–110. Springer.

[Borgida* 10] Borgida, A., de Bruijn, J., Franconi, E., Seylan, I., Straccia, U.,
On finding query rewritings under expressive constraints.
In Proc. 18th Italian Symp. on Advanced Database Systems, SEBD 2010.
   Asking queries about frames.
   Morgan Kaufmann.

   View-based query processing: On the relationship between rewriting, answering and losslessness.

   Modular reuse of ontologies: Theory and practice.

   Decomposable negation normal form.
Structure identification in relational data.
AI, 58:237–270.

Computing circumscription revisited: A reduction algorithm.

Computing strongest necessary and weakest sufficient conditions of first-order formulas.
In IJCAI-01, pages 145–151. Morgan Kaufmann.

Effective preprocessing in SAT through variable and clause elimination.
In SAT 2005, volume 3569 of LNCS, pages 61–75.

Stable models and circumscription.
AI, 175(1):236–263.
Exact query reformulation over databases with first-order and description logics
ontologies.

Quantifier elimination in second-order predicate logic.

Second-Order Quantifier Elimination: Foundations, Computational Aspects and
Applications.

Did I damage my ontology? A case for conservative extensions in description
logics.

Answering queries using views: a survey.
Clause elimination procedures for CNF formulas.

Unfolding partiality and disjunctions in stable model semantics.

The role of abduction in logic programming.

Forgetting and uniform interpolation in large-scale description logic terminologies.

Uniform interpolation of *ALC*-ontologies using fixpoints.
Propositional independence – formula-variable independence and forgetting.
*JAIR*, 18:391–443.

Circumscription.

A *Study of Nonmonotonic Reasoning*.

On strongest necessary and weakest sufficient conditions.

Abduction in logic programming: A new definition and an abductive procedure based on rewriting.
*AI*, 140(1/2):175–205.
   Soundness of inprocessing in clause sharing SAT solvers.

   Queries determined by views: pack your views.
   In PODS ’07, pages 23–30. ACM.

   Circumscription – a form of non-monotonic reasoning.

   Views and queries: Determinacy and rewriting.
   TODS, 35(3).

   Well-founded semantics coincides with three-valued stable semantics.
   Fundamenta Informaticae, 13(4):445–464.
Views and queries: Determinacy and rewriting.
In PODS 2005, pages 49–60.

Knowledge compilation using Horn approximations.

Einige methologische Untersuchungen zur Definierbarkeit der Begriffe.
Erkenntnis, 5:80–100.

Literal projection for first-order logic.
In JELIA 08, volume 5293 of LNCS (LNAI), pages 389–402. Springer.

Circumscription and projection as primitives of logic programming.
In Tech. Comm. ICLP’10, volume 7 of LIPIcs, pages 202–211.
    Projection and scope-determined circumscription.
    *JSC*, 47(9):1089–1108.

    Abduction in logic programming as second-order quantifier elimination.

    Computing with logic as operator elimination: The ToyElim system.
    In *INAP 2011/WLP 2011*, volume 7773 of *LNCS (LNAI)*. Springer.

    Expressing view-based query processing and related approaches with second-order operators.
Notes on the Relationship to Craig Interpolation (Addendum to Slide 9)

- [Tarski 35]: **Definability w.r.t. first-order formulas can be reduced to first-order validity**

\[ \text{gsnc}_S(F, G) \models \text{gwsc}_S(F, G) \iff F \land G \models F' \rightarrow G' \]

- The **interpolants** \( X \) in \( S \) such that

\[ F \land G \models X \models F' \rightarrow G' \]

are definitions

- The extreme definitions GSNC and GWSC are obtained as **uniform interpolants** – if the predicate elimination succeeds

More precisely: Let \( S \) specify a set of predicates. Let \( F, G \) be first-order. Let \( F', G' \) be \( F, G \) after systematically replacing all predicates not in \( S \) with new symbols. Then

\[ \text{gsnc}_S(F, G) \models \text{gwsc}_S(F, G) \iff F \land G \models F' \rightarrow G'. \]

If \( X \subseteq S \), then \( F \land G \models X \iff \text{gsnc}_S(F, G) \models X \).

If \( X \subseteq S \), then \( X \models F' \rightarrow G' \iff X \models \text{gwsc}_S(F, G) \).
Notes About Unique Definability (Mentioned on Slides 10 and 14)

- If $S \equiv \overline{S}$, then a formula that is definable in terms of $S$ within $F$ is uniquely definable iff

$$\models \text{project}_S(F)$$

- **Conservativeness** with respect to all formulas in a scope and definability in terms of that scope together imply **unique definability**

See [W 14a]
Proof Sketch for Slide 10

Assumptions: \( R \subseteq U, Q \subseteq D \)

\( R \) is an exact rewriting of \( Q \) w.r.t. \( V \)

iff \( \forall DB \subseteq D: \text{project}_U(V \land DB) \models R \) iff \( DB \models Q \)

iff \( \forall DB \subseteq D: V \land DB \models R \) iff \( DB \models Q \) since \( R \subseteq U \)

iff \( \forall DB \subseteq D: DB \models \neg V \lor R \) iff \( DB \models Q \)

iff \( \text{project}_D(V \land \neg R) \equiv \text{project}_D(\neg Q) \)

iff \( \text{gwsc}_D(V, R) \equiv Q \). since \( Q \subseteq D \)

Assume A1: Unique definability of all \( R \subseteq U \) i.t.o. \( D \) within \( V \), i.e.

\( \forall R \subseteq U: \text{gsnc}_D(V, R) \equiv \text{gwsc}_D(V, R). \)

\( \text{gwsc}_D(V, R) \models Q \)

iff \( \text{gsnc}_D(V, R) \models Q \) by assumption A1

iff \( V \land R \models Q \) since \( Q \subseteq D \)

iff \( V \land \neg Q \models \neg R \)

iff \( \text{project}_U(V \land \neg Q) \models \neg R \) since \( R \subseteq D \)

iff \( R \models \text{gwsc}_U(V, Q) \). Note: for “sound views” just this direction is relevant

\( Q \models \text{gwsc}_D(V, R) \)

iff \( \text{project}_D(V \land \neg R) \models \neg Q \)

iff \( V \land \neg R \models \neg Q \) since \( Q \subseteq D \)

iff \( V \land Q \models R \) since \( R \subseteq U \)

iff \( \text{gsnc}_U(V, Q) \models R \). since \( R \subseteq U \)

See [W 14a]
Proof Sketch for Slide 11

Assumption: $R \subseteq D$

$R$ is a split rewriting of $Q$ w.r.t. $V$ and $D$

iff $\forall DB \subseteq D : DB \models R$ iff $DB \land V \models Q$

iff $\forall DB \subseteq D : DB \models R$ iff $DB \models \neg V \lor Q$

iff $\text{project}_D(\neg R) \equiv \text{project}_D(V \land \neg Q)$

iff $\neg R \equiv \text{project}_D(V \land \neg Q)$ since $R \subseteq D$

iff $R \equiv \text{gwsc}_D(V, Q)$.

- Note: The GWSC is the **only** solution!
- This seems to supersede material in [W 14a]
Proof Sketch for Slide 15

\[ \models \forall x F \land (x \leftrightarrow X) \rightarrow (G \leftrightarrow H) \]

iff \[ \models (\forall x F \land x \land X \rightarrow (G \leftrightarrow H)) \land (\forall x F \land \neg x \land \neg X \rightarrow (G \leftrightarrow H)) \]

iff \[ \models (X \rightarrow (\forall x F \land x \rightarrow (G \leftrightarrow H))) \land ((\exists x F \land \neg x \land \neg (G \leftrightarrow H)) \rightarrow X) \]

iff \[ X \models \forall x F \land x \rightarrow (G \leftrightarrow H) \text{ and } \exists x F \land \neg x \land \neg (G \leftrightarrow H) \models X. \]