WEAK EQUIVALENCE: THEORY AND APPLICATIONS

Karen L. Kwast
Sieger van Denneheuvel

ILLC Prepublication Series
for Computation and Complexity Theory CT-92-02

The ILLC Prepublication Series

1990

Logic, Semantics and Philosophy of Language

LP-90-01 Jaap van der Does — A Generalized Quantifier Logic for Naked Infinitives
LP-90-02 Jeroen Groenendijk, Martin Stokhof — Dynamic Montague Grammar
LP-90-03 Renate Bartsch — Concept Formation and Concept Composition
LP-90-04 Aarne Ranta — Intuitionistic Categorial Grammar
LP-90-05 Patrick Blackburn — Nominal Tense Logic
LP-90-06 Gennaro Chierchia — The Variablity of Impersonal Subjects
LP-90-07 Gennaro Chierchia — Anaphora and Dynamic Logic
LP-90-08 Herman Hendriks — Flexible Montague Grammar
LP-90-09 Paul Dekker — The Scope of Negation in Discourse, towards a Flexible Dynamic Montague grammar
LP-90-10 Theo M.V. Janssen — Models for Discourse Markers
LP-90-11 Johan van Benthem — General Dynamics
LP-90-12 Serge Lapierre — A Functional Partial Semantics for Intensional Logic
LP-90-13 Zhisheng Huang — Logics for Belief Dependence
LP-90-14 Jeroen Groenendijk, Martin Stokhof — Two Theories of Dynamic Semantics
LP-90-15 Maarten de Rijke — The Modal Logic of Inequality
LP-90-16 Zhisheng Huang, Karen Kwast — Awareness, Negation and Logical Omniscience
LP-90-17 Paul Dekker — Existential Disclosure, Implicit Arguments in Dynamic Semantics

Mathematical Logic and Foundations

ML-90-01 Harold Schellinx — Isomorphisms and Non-Isomorphisms of Graph Models
ML-90-02 Jaap van Oosten — A Semantical Proof of De Jongh's Theorem
ML-90-03 Yde Venema — Relational Games
ML-90-04 Maarten de Rijke — Unary Interpretability Logic
ML-90-05 Domenico Zambella — Sequences with Simple Initial Segments
ML-90-06 Jaap van Oosten — Extension of Lifschitz' Realizability to Higher Order Arithmetic, and a Solution to a Problem of F. Richman
ML-90-07 Maarten de Rijke — A Note on the Interpretability Logic of Finitely Axiomatized Theories
ML-90-08 Harold Schellinx — Some Syntactical Observations on Linear Logic
ML-90-09 Dick de Jongh, Duccio Pianigiani — Solution of a Problem of David Guaspari
ML-90-10 Michiel van Lambalgen — Randomness in Set Theory
ML-90-11 Paul C. Gilmore — The Consistency of an Extended NaDSet

Computation and Complexity Theory

CT-90-01 John Tromp, Peter van Emde Boas — Associative Storage Modification Machines
CT-90-02 Sieger van Denneheuvel, Gerard R. Renardel de Lavalette — A Normal Form for PCSJ Expressions
CT-90-03 Ricard Gavaldà, Leen Torenvliet, Osamu Watanabe, José L. Balcázar — Generalized Kolmogorov Complexity in Relativized Separations
CT-90-04 Harry Buhrman, Edith Spaan, Leen Torenvliet — Bounded Reductions
CT-90-05 Sieger van Denneheuvel, Karen Kwast — Efficient Normalization of Database and Constraint Expressions
CT-90-06 Michiel Smid, Peter van Emde Boas — Dynamic Data Structures on Multiple Storage Media, a Tutorial
CT-90-07 Kees Doets — Greatest Fixed Points of Logic Programs
CT-90-08 Fred de Geus, Ernest Rotterdam, Sieger van Denneheuvel, Peter van Emde Boas — Physiological Modelling using RL
CT-90-09 Roel de Vrijer — Unique Normal Forms for Combinatory Logic with Parallel Conditional, a case study in conditional rewriting

Other Prepublications

X-90-01 A.S. Troelstra — Remarks on Intuitionism and the Philosophy of Mathematics, Revised Version
X-90-02 Maarten de Rijke — Some Chapters on Interpretability Logic
X-90-03 L.D. Beklemishev — On the Complexity of Arithmetical Interpretations of Modal Formulae
X-90-04 — Annual Report 1989
X-90-05 Valentin Shehtman — Derived Sets in Euclidean Spaces and Modal Logic
X-90-06 Valentin Goranko, Solomon Passy — Using the Universal Modality: Gains and Questions
X-90-07 V.Yu. Shavrukov — The Lindenbaum Fixed Point Algebra is Undecidable
X-90-08 L.D. Beklemishev — Provability Logics for Natural Turing Progressions of Arithmetical Theories
X-90-09 V.Yu. Shavrukov — On Rosser's Provability Predicate
X-90-10 Sieger van Denneheuvel, Peter van Emde Boas — An Overview of the Rule Language RL/1
X-90-11 Alessandra Carbone — Provable Fixed points in $I\Delta_0+\Omega_1$, revised version
X-90-12 Maarten de Rijke — Bi-Unary Interpretability Logic
X-90-13 K.N. Ignatiev — Dzhaparidze's Polymodal Logic: Arithmetical Completeness, Fixed Point Property, Craig's Property
X-90-14 L.A. Chagrova — Undecidable Problems in Correspondence Theory
X-90-15 A.S. Troelstra — Lectures on Linear Logic

1991

Logic, Semantics and Philosophy of Langauge

LP-91-01 Wiebe van der Hoek, Maarten de Rijke — Generalized Quantifiers and Modal Logic
LP-91-02 Frank Veltman — Defaults in Update Semantics
LP-91-03 Willem Groeneveld — Dynamic Semantics and Circular Propositions
LP-91-04 Makoto Kanazawa — The Lambek Calculus enriched with Additional Connectives
LP-91-05 Zhisheng Huang, Peter van Emde Boas — The Schoenmakers Paradox: Its Solution in a Belief Dependence Framework
LP-91-06 Zhisheng Huang, Peter van Emde Boas — Belief Dependence, Revision and Persistence
LP-91-07 Henk Verkuyl, Jaap van der Does — The Semantics of Plural Noun Phrases
LP-91-08 Víctor Sánchez Valencia — Categorial Grammar and Natural Reasoning
LP-91-09 Arthur Nieuwendijk — Semantics and Comparative Logic
LP-91-10 Johan van Benthem — Logic and the Flow of Information

Mathematical Logic and Foundations

ML-91-01 Yde Venema — Cylindric Modal Logic
ML-91-02 Alessandro Berarducci, Rineke Verbrugge — On the Metamathematics of Weak Theories
ML-91-03 Domenico Zambella — On the Proofs of Arithmetical Completeness for Interpretability Logic
ML-91-04 Raymond Hoofman, Harold Schellinx — Collapsing Graph Models by Preorders

Institute for Logic, Language and Computation
Plantage Muidergracht 24
1018TV Amsterdam
Telephone 020-525.6051, Fax: 020-525.5101

WEAK EQUIVALENCE: THEORY AND APPLICATIONS

Karen L. Kwast
Sieger van Denneheuvel
Department of Mathematics and Computer Science
University of Amsterdam

ILLC Prepublications
for Computation and Complexity Theory
ISSN 0924-8374

Coordinating editor: Dick de Jongh

received July 1992

Weak Implication: Theory and Applications

Karen L. Kwast & Sieger van Denneheuvel[*]
University of Amsterdam
Department of Mathematics & Computer Science,
Plantage Muidergracht 24, 1018 TV, Amsterdam.

Abstract

We study a generalization of the classical notion of implication, called *weak implication.* It extends unquantified predicate logic with a single level of *existential quantification.* We present a sound and complete set of deduction rules for weak implications. The notion of weak implication was introduced for the sake of a formal specification of a *symbolic constraint solving* system. Other practical *applications* of can be found in the realm of relational database theory: query normalization and integrity constraints in the context of views.

[*]Present address: Sy*ll*ogic BV, postbus 26, 3990 DA, Houten, NL.

Contents

1 Introduction

Weak equivalence is a generalization of the classical notion of equivalence. Two formulas are equivalent if they are satisfied by the same values; they are weakly equivalent if they are equivalent 'on' a set of variables X, that is, if their projections on X are equivalent. Consider for instance the non-equivalent constraints

$$x = y + 2,\ y = z + 3 \ \not\equiv\ x = z + 5 \tag{1}$$

If one wants to express x in terms of the known variable z, it makes sense to ask for the strongest condition on x and z that is entailed by the left-hand side. The actual value of y is irrelevant and becomes existentially quantified. Formally:

$$x = y + 2,\ y = z + 3 \ \equiv_{xz}\ x = z + 5 \tag{2}$$

A more interesting example derives from the context of constraint solving. The following set of constraints is underdetermined:

$$x + u = v,\ y + u = v,\ x + y = 6m,\ u < v \tag{3}$$

One can solve x and y in terms of m by means of the equivalent$_{mxy}$ constraints

$$x = 3 * m,\ y = 3 * m,\ m > 0 \tag{4}$$

Obviously, these 2 sets of constraints are not equivalent, but they express the same relation between x, y and m.

In this paper, we present the logical theory of weak implications. § 2 summarizes related research. Basic notions and definitions are provided in § 3. The system $\mathcal{WI}$ of derivation rules for weak implications is discussed in § 4. Applications are given in § 5: constraint solving, § 6: query normalization and § 7: integrity constraints.

2 Related Work

There are several areas of research that are related to the present subject. Obviously, the present system will be a subsystem of first order predicate logic, namely the logic of $\forall\exists$-sentences. This reduction in expressive power will be compensated for by an increase in transparency: the absence of (nested) quantifiers. Moreover, this fragment is known to be decidable.

Restrictions to the language are not uncommon: in logic programming (e.g. [1]), the language is restricted to unquantified predicate logic with equality and functions. *All* free variables are (implicitly and globally) universally quantified. Here we add in effect a second layer of quantification: (local) existential quantification.

In unification theory (see [7]), one studies the possibility to find a *most general unifier* Φ for a set of equations Γ in the context of an equational theory E. One can check Φ in terms of equivalence: $E \models \Gamma[\Phi] \equiv \text{TRUE}$, or, equivalently, $E \models \Gamma \equiv \Phi$. Here we study the more general issue: $\Gamma \equiv_X \Delta$ (in the context of E), where Φ may be part of Δ.

The present theory originates from yet another area: symbolic constraint solving. A good example is the system *Mathematica* ([9]). It allows one to declare a set of constraints, which can be simplified by the system. Moreover, one can ask for a symbolic solution, expressing some *wanted* variables in terms of the others (: the *known* variables). When the set of constraints (such as (3) above) is underdetermined, the system will fail.

Weak equivalence has been developed as a tool to describe and validate the **RL/1** symbolic constraint solving system, which allows for the specification of *intermediate* variables. This system is capable of producing a *conditional* solution: if the known variables satisfy some condition, then they define the wanted variables in the specified manner (cf. (4) above). We will illustrate this idea in § 5 by means of an example; for all details on **RL/1** see [2].

The system **RL/1** has much in common with **CLP** ([5]), where global constraints are combined with local conditions into rules and goals (: as in PROLOG). If a goal is satisfiable, a successful derivation will yield a set of *answer constraints* as *symbolic output*. In **CLP** the answer constraints are represented in *solved form*, which depends on the type of constraints. In **RL/1**, however, the answer consists of a *reduced constraint* and a *solution set*, that is, a symbolic solution expressing some variables in terms of others (see § 5). This has the advantage that the symbolic answer can be evaluated on a (relational) database, in accordance with the **RL** objective to integrate logic programming with database systems. On the other hand, it restricts the types of constraints that can be dealt with; **RL/1** does not cover recursive rules.

In § 5 we employ weak equivalence to give a formal specification of the **RL/1** constraint solver; its technical details and a comparison to other systems for constraint solving can be found in [2].

There are many other possible applications of the notion of weak implication. It can be applied to study the implication problem of any phenomenon that requires but a single level of existential quantification. In the present formalism, all quantifiers are removed and replaced by implicit quantification, by means of the implication variables. These variables connect the antecedent of the implication with the consequence and are (implicitly) universally quantified. All remaining variables are more or less *irrelevant*: they are (implicitly) existentially quantified, locally, that is, the scope of quantification is restricted to antecedent cq consequence.

The advantage of the weak implication formalism is that it corresponds with the algebraic projection operator: the projection set contains the relevant variables, all others can be ignored (: existentially quantified). As a consequence, weak implication statements are more intuitive and transparent, at least to people that are used to the relational algebra.

Weak implications can be applied to *normalize* terms of the relational algebra, highlighting the natural boundaries of the projection normal form, $\Pi_X \sigma_\varphi(R \bowtie S)$, which corresponds with the basic **SQL** statement, `SELECT` X `FROM` R, S `WHERE` φ. Normalization theory does not *require* weak implications, of course, but its description can be simplified by means of weakly equivalent (sub)terms.

In a similar manner, weak implication can be applied to derive integrity constraints. There exist several formalisms to study the integrity implication problems, but none that includes views and equivalent term rewriting in a uniform manner.

The applications we give of the notion of weak implication are mainly illustrative. The results on term rewriting are freely adapted from previous work on term rewriting, by ourselves ([4]) as well as by others (e.g. [10]), only 'translated' into the weak implication formalism to illustrate the usefulness of the latter. The examples of derived integrity constraints have not yet been systematically developed; for the basics of database theory we refer to [8].

3 Definitions

Let $\mathcal{L}$ be a predicate language, consisting of constants CON $(a, b, c, \ldots)$, variables VAR $(x, y, z, \ldots)$, predicates $(R, S, T, \ldots)$ and functions $(f, g, \ldots)$. Terms $(s, t, \ldots)$ and formulas FORM $(\varphi, \psi, \ldots)$ are constructed as usual. $\mathcal{L}$ contains an identity symbol $=$ and the propositional constants TRUE and FALSE. $\alpha(\varphi)$ is the set of free variables of a formula φ. Attributes $\mathcal{A}$ and variables will be used indifferently; sets of attributes are denoted by X or $\mathbf{x}$.

A *solution* is an equation $x = t$ with $x \notin \alpha(t)$. A *solution set* (: $\Phi, \Psi, \ldots$) is a finite set of independent solutions, that is:

Definition 1 *A solution set Φ is a finite set $\{ x_1 = t_1, \ldots, x_n = t_n \}$, such that for all $i, j \leq n$: $x_i \neq x_j$ $(i \neq j)$ and $x_i \notin \alpha(t_j)$.*

The syntactic independence restriction on a solution set Φ can be reformulated in terms of its *head- and tail attributes*:

Definition 2 $\alpha_H(\Phi) := \{x_1, \ldots, x_n\} \quad \alpha_T(\Phi) := \alpha(t_1) \cup \ldots \cup \alpha(t_n)$.

The set $\Phi = \{x_1 = t_1, \ldots, x_n = t_n\}$ is a solution set iff the number of head attributes is n, that is, $\#(\alpha_H(\Phi)) = \#(\Phi)$, and $\alpha_H(\Phi) \cap \alpha_T(\Phi) = \emptyset$. Considered as a substitution, Φ is separated on its head attributes away from its tail attributes.

Let $\mathcal{M} = \langle \mathcal{D}, \mathcal{I} \rangle$ be a model for a language $\mathcal{L}$ and $\mathbf{H}$ the set of assignments $h : \text{VAR} \to \mathcal{D}$. $\mathcal{M} \models \varphi[h]$ is defined by the usual induction. Obviously, if $h =_{\alpha(\varphi)} h'$, that is, if $h(x) = h'(x)$ for all $x \in \alpha(\varphi)$, then $\mathcal{M} \models \varphi[h]$ iff $\mathcal{M} \models \varphi[h']$.

The notion of a valid implication is completely standard: $\varphi \models \psi$ iff every model and assignment that satisfies φ satisfies ψ as well. It is called *strong* here to discriminate it from its weaker generalization.

Weak implication is more general in the sense that it includes strong implication as a special case, but it is weaker in the sense that any weak implication is logically implied by its strong counterpart. Before we give the formal definition, an example will illustrate this point.

Example 1 *Some strong and weak implications.*

$$\begin{array}{llll}
x = y + 1 \models x > y & & x > y \not\models x = y + 1 \\
x = y + 1 \models_{xy} x > y & & x > y \not\models_{xy} x = y + 1 \\
x = y + 1 \models_{x} x > y & & x > y \models_{x} x = y + 1
\end{array}$$

Weak implication is defined on model-level as well as in general:

Definition 3 *Weak implication*

1. $\mathcal{M} : \varphi \models_X \psi \;:=\; \forall h \in \mathbf{H} : \mathcal{M} \models \varphi[h] \Rightarrow \exists h' \in \mathbf{H} : h' =_X h \;\&\; \mathcal{M} \models \psi[h']$

2. $\varphi \models_X \psi \;:=\;$ *for all models* $\mathcal{M}$: $\mathcal{M} : \varphi \models_X \psi$

The relevance of the notion $\mathcal{M} : \varphi \models_X \psi$ can be explained by means of an example:

$$x = -1 \models_x x = y^2 \wedge x + 1 = 0$$

It depends on the model (: is $\sqrt{-1}$ defined?) whether or not there exists a satisfying assignment h'. As we do *not* want to fix the models from the outset, we *must* be able to conceive weak implications as a notion valid in a model $\mathcal{M}$ or in a class of models $\mathcal{C}$. Note that we do not define weak implication on assignment-level, that is, as a language connective. To see why, suppose we would define a connective $\supset_X$.

Definition 4 *Weak connective*
$\mathcal{M} \models \varphi \supset_X \psi[h] \;:=\; \mathcal{M} \models \varphi[h] \;\Rightarrow\; \exists h' \in \mathbf{H} : h' =_X h \;\&\; \mathcal{M} \models \psi[h']$

Given this definition we would derive at a system in which even **transitivity** is unvalid: $\varphi \supset_X \psi,\; \psi \supset_X \chi \;\not\models\; \varphi \supset_X \chi$.

Example 2 *Suppose* $h(x) = 3, h(y) = 5, h(z) = 2$, *then*
$\mathcal{M} \models x = 3 \supset_{xy} y = 5 * z\ [h]$
$\mathcal{M} \models y = 5 * z \supset_{xy} x = 11\ [h]$
$\mathcal{M} \not\models x = 3 \supset_{xy} x = 11\ [h]$

Note that this is not a counterexample under a higher level reading; the rule

$$\varphi \models_X \psi,\; \psi \models_X \chi \;\Rightarrow\; \varphi \models_X \chi$$

is valid, but most models will falsify the second assumption (: $y = 5 * z \models_{xy} x = 11$), thus avoiding the undesirable consequence (: $x = 3 \models_{xy} x = 11$).

Definition 5 $\varphi \equiv_X \psi \;:=\; \varphi \models_X \psi \;\&\; \psi \models_X \varphi$

Before we turn to the logic of weak implications, we will try to demystify this notion by exploring its translation into quantified predicate formulas.

3.1 Clashing Variables

The major effect of indexing implication with a set of attributes $\mathbf{x}$ is that the remaining attributes (: $\mathbf{y}$ and $\mathbf{z}$ respectively) get existentially quantified. Hence:

Lemma 1 *Let* $\mathbf{y} = \alpha(\varphi) \setminus \mathbf{x}$ *and* $\mathbf{z} = \alpha(\psi) \setminus \mathbf{x}$, *then*
$\mathcal{M} : \varphi \models_{\mathbf{x}} \psi$ *iff* $\mathcal{M} \models \forall\mathbf{x}(\exists\mathbf{y}\ \varphi(\mathbf{xy}) \supset \exists\mathbf{z}\ \psi(\mathbf{xz}))$

Example 3 $m < x \;\models_x\; x = m + n \wedge n > 0$
corresponds with $\forall x(\exists m :\ m < x \supset \exists m, n :\ x = m + n \wedge n > 0)$

In general a weak implication may contain *clashing variables* (such as: m in the example above), that is, variables that appear on both sides of the implication sign but not in X. If that is the case, an implication must be *rectified* or at least *purified*, before it can be transformed into $\forall\exists$-normal form.

Definition 6 *An implication* $\varphi \models_X \psi$ *is rectified iff* $\alpha(\varphi) \subseteq X$.

Strong implication is a special case of weak implication: put $X = \mathcal{A}$, the set of *all* (relevant) attributes, that is: $\varphi \models \psi$ iff $\varphi \models_{\mathcal{A}} \psi$.

As a consequence, every strong implication is rectified. Moreover, every rectified *equivalence* is bound to be strong (cf. the rule **irrelevance**, see below). Hence we prefer a somewhat weaker property:

Definition 7 *An implication $\varphi \models_X \psi$ is purified iff $\alpha(\varphi) \cap \alpha(\psi) \subseteq X$.*

The absence of clashing variables guarantees that all variables that are shared by φ and ψ must be relevant (: in X).

Example 4 *Compare:*
$m < x \models_{xm} x = m + n \wedge n > 0$ *is rectified and purified,*
$m < x \models_{xm} x = k + n \wedge n > 0$ *is rectified and purified,*
$m < x \models_{x} x = k + n \wedge n > 0$ *is purified.*

Every implication can be transformed into an equivalent purified or rectified implication. On account of the confusing outlook of $x < y \models_x x > y$ it is advisable to purify an implication as soon as possible. Moreover, the corresponding predicate formula can be brought into $\forall\exists$-normal form, that is, a sentence with prefix of the format $\forall\exists$ and unquantified prenex.

Lemma 2 *Let $\mathbf{y} = \alpha(\varphi) \setminus \mathbf{x}$, $\mathbf{z} = \alpha(\psi) \setminus \mathbf{x}$ and $\mathbf{y} \cap \mathbf{z} = \emptyset$ (: purified), then:*
$\mathcal{M} : \varphi \models_{\mathbf{x}} \psi$ *iff* $\mathcal{M} \models \forall \mathbf{xy} \exists \mathbf{z}(\varphi(\mathbf{xy}) \supset \psi(\mathbf{xz}))$

(Note the change in quantifier for $\mathbf{y}$; cf. universally quantified Horn clauses.)

Example 5 *The weak implications of example 4 correspond with, respectively:*
$\forall x, m\, \exists n(m < x \supset x = m + n \wedge n > 0)$,
$\forall x, m\, \exists k, n(m < x \supset x = k + n \wedge n > 0)$,
$\forall x\, \exists m, k, n(m < x \supset x = k + n \wedge n > 0)$.

3.2 Substitution

In the sequel we will employ solution sets as *substitutions*. The notation $\varphi[\Phi]$ refers to the *result* of the substitution Φ on φ. Any solution set can be used as a substitution, replacing the head variables of Φ in φ by the corresponding tails. Note that the result is well-defined, on account of the well-formedness conditions on solution sets, which guarantee that the individual substitutions are independent.

Substitution has some well-known properties, which will be used here without further justification. In particular:

Lemma 3 *For all formulas φ, ψ and every solution set Φ:*
1. $\alpha_H(\Phi) \cap \alpha(\varphi[\Phi]) = \emptyset$.
2. If $\alpha(\varphi) \cap \alpha_H(\Phi) = \emptyset$, *then* $\varphi[\Phi] = \varphi$.
3. If $\Phi = \Phi_1 \cup \Phi_2$, *then* $\varphi[\Phi] = \varphi[\Phi_1][\Phi_2] = \varphi[\Phi_2][\Phi_1]$.
4. If $\Phi =_{\alpha(\varphi)} \Psi$, *then* $\varphi[\Phi] = \varphi[\Psi]$.
5. $\Phi[\Phi] \equiv \text{TRUE}$.

Any effective procedure that computes $\varphi[\Phi]$ from φ and Φ satisfies these properties. A trivial example of (3.5) is: $x = 8[x = 8]$ $(:= 8 = 8) \equiv \text{TRUE}$.

4 Rules for Weak Implication

The inference rules for strong implications can be generalized for weak implications. A list of them is given in figure 1: the system $\mathcal{WI}$ of derivation rules for weak implications. Figure 2 contains some derived rules and meta-rules.

All rules in figure 1, except for **abstraction**, are in fact strong, which means that these rules hold for strong implications as well as for arbitrary weak implications.

Some restrictions are placed on the meta-rules, since they are not sound unrestrictedly. This can be shown by means of examples; for instance, to explain the restriction on **union**:

Example 6 *Both* $x < 3 \models_x x < y \wedge y = 4$ *and* $x < 3 \models_x x < y \wedge y = 5$.
Still, $x < 3 \not\models_x x < y \wedge y = 4 \wedge y = 5$, *since* $(x < y \wedge y = 4 \wedge y = 5) \equiv \text{FALSE}$.

Similar examples can be given for all restrictions (see [6]).

Contraposition is de facto only valid for strong implications (: on account of the **strong** rule), but this is compensated for by the general validity of **absurdum**.

A rule worth noticing is **all cases**. Whereas **union** is restricted to compatible consequences, **all cases** is valid for all pairs of antecedents. In particular, putting $\varphi := \varphi \wedge \psi$ and $\phi := \varphi \wedge \neg\psi$, it implies the derived rule **either or**.

The **cut** rule is only applicable if the untrimmed implication is non-trivial. Another remarkable rule is **implication**. Its validity is restricted to antecedents with explicitly mentioned variables.

Theorem 4 *The rules and meta-rules of system $\mathcal{WI}$ (: figure 1) are all sound.*

Proof: Straightforward; see [6].
■

Theorem 5 *All rules in figure 2 can be derived from $\mathcal{WI}$.*

Proof: We will only prove **instantiation**, by way of example.
To be proven: $\varphi \models_X \psi \Rightarrow \varphi[\Phi] \models_X \psi[\Phi]$, where $\alpha(\Phi) \cap \alpha(\psi) \subseteq X$.
Define $Y := \alpha(\varphi[\Phi] \wedge \psi[\Phi])$, which implies $Y \cap \alpha_H(\Phi) = \emptyset$.

1	given	$\varphi \models_X \psi$	
2	weakening	$\varphi \wedge \Phi \models_X \varphi$	
3	transitivity: 2,1	$\varphi \wedge \Phi \models_X \psi$	
4	weakening	$\varphi \wedge \Phi \models_X \Phi$	
5	union: 3,4	$\varphi \wedge \Phi \models_X \psi \wedge \Phi$	(: $\alpha(\Phi) \cap \alpha(\psi) \subseteq X$)
6	removal: 5	$\varphi \wedge \Phi \models_{X \cap Y} \psi \wedge \Phi$	
7	abstraction	$\varphi[\Phi] \models_Y \varphi \wedge \Phi$	(: $Y \cap \alpha_H(\Phi) = \emptyset$)
8	removal: 7	$\varphi[\Phi] \models_{X \cap Y} \varphi \wedge \Phi$	
9	transitivity: 8,6	$\varphi[\Phi] \models_{X \cap Y} \psi \wedge \Phi$	
10	substitution: 9	$\varphi[\Phi] \models_{X \cap Y} \psi[\Phi]$	
11	irrelevance: 10	$\varphi[\Phi] \models_X \psi[\Phi]$	(: $\alpha(\psi[\Phi]) \subseteq Y$)

For strong implications the restriction is trivial, so: $\varphi \models \psi \Rightarrow \varphi[\Phi] \models \psi[\Phi]$.
■

Name	Rule	Restriction
true	$\varphi \models_X \text{TRUE}$	
false	$\varphi \wedge \neg\varphi \models_X \text{FALSE}$	
weakening	$\varphi \wedge \psi \models_X \varphi \vee \chi$	
duality	$\neg\varphi \wedge \neg\psi \models_X \neg(\varphi \vee \psi)$	
substitution	$\varphi \wedge \Phi \models_X \varphi[\Phi]$	
generalization	$\varphi[\Phi] \wedge \Phi \models_X \varphi \wedge \Phi$	
abstraction	$\varphi[\Phi] \models_X \varphi \wedge \Phi$	$\alpha_H(\Phi) \cap X = \emptyset$
removal	$\varphi \models_{XY} \psi \Rightarrow \varphi \models_X \psi$	
irrelevance	$\varphi \models_X \psi \Rightarrow \varphi \models_{XY} \psi$	$Y \cap \alpha(\psi) \subseteq X$
transitivity	$\varphi \models_X \psi,\ \psi \models_X \chi \Rightarrow \varphi \models_X \chi$	
union	$\varphi \models_X \psi,\ \varphi \models_X \chi \Rightarrow \varphi \models_X \psi \wedge \chi$	$\alpha(\psi) \cap \alpha(\chi) \subseteq X$
all cases	$\varphi \models_X \psi,\ \phi \models_X \psi \Rightarrow \varphi \vee \phi \models_X \psi$	
contraposition	$\neg\varphi \models_X \neg\psi \Rightarrow \psi \models_X \varphi$	$\alpha(\psi) \subseteq X$

Figure 1: Rules and meta-rules of $\mathcal{WI}$.

Name	Rule	Restriction
tertium	$\text{TRUE} \models_X \varphi \vee \neg\varphi$	
absurdum	$\text{FALSE} \models_X \varphi$	
independence	$\text{TRUE} \models_\emptyset \Phi$	
Leibniz	$s = t \wedge \varphi(s) \models \varphi(t)$	
strong	$\varphi \models_X \psi \Rightarrow \varphi \models \psi$	$\alpha(\psi) \subseteq X$
instance	$\text{TRUE} \models_X \varphi(y) \Rightarrow \text{TRUE} \models_X \varphi(a)$	$y \in X$
instantiation	$\varphi \models_X \psi \Rightarrow \varphi[\Phi] \models_X \psi[\Phi]$	$\alpha(\Phi) \cap \alpha(\psi) \subseteq X$
MP	$\text{TRUE} \models_X \varphi,\ \varphi \models_X \psi \Rightarrow \text{TRUE} \models_X \psi$	
MT	$\text{TRUE} \models \neg\psi,\ \varphi \models_X \psi \Rightarrow \text{TRUE} \models \neg\varphi$	
augmentation	$\varphi \models_X \psi \Rightarrow \varphi \wedge \phi \models_X \psi \wedge \phi$	$\alpha(\phi) \cap \alpha(\psi) \subseteq X$
implication	$\phi \models_X \varphi \supset \psi \Rightarrow \phi \wedge \varphi \models_X \psi$	$\alpha(\varphi) \subseteq X$
deduction	$\phi \wedge \varphi \models_X \psi \Rightarrow \phi \models_X \varphi \supset \psi$	
either or	$\varphi \wedge \psi \models_X \chi,\ \varphi \wedge \neg\psi \models_X \chi \Rightarrow \varphi \models_X \chi$	
reduction	$\varphi \wedge \psi \models_X \neg\psi \Rightarrow \varphi \models_X \neg\psi$	
cut	$\varphi \wedge \psi \models_X \chi, \varphi \models_X \psi \Rightarrow \varphi \models_X \chi$	$\alpha(\varphi) \cap \alpha(\psi) \subseteq X$

Figure 2: Derived rules of the system $\mathcal{WI}$

4.1 Propositional Logic

At first glance, $\mathcal{WI}$ can be partitioned into 3 parts: propositional rules (8), "quantification" rules to vary the implication variables (2) and rules dealing with identity in terms of solution sets (3). However, abstraction is in essence a quantification rule: the quantified analogon of $\varphi[x = c] \models_Y \varphi$ ($x \notin Y$) is the $\exists$-introduction rule $\varphi(c) \models \exists x\ \varphi(x)$.

The rules in system $\mathcal{WI}$ are still incomplete in the sense that basic properties of the propositional connectives are not listed, namely those that derive from basic set theory, such as $X = X \cup X$ and $X \cup Y = Y \cup X$. In particular, we need thinning and permutation rules (: $\vee$ and $\wedge$ are idempotent, symmetric and associative) and the law of double negation (: $\neg\neg\varphi \equiv \varphi$).

The rules in $\mathcal{WI}$ have been choosen to pinpoint where weak implications differ from strong ones. It should be obvious that all propositional valid inferences are derivable by the present system.

4.2 System $\mathcal{WI}$ is Complete

Let $\vdash_X$ be the smallest relation over unquantified formulas and sets of variables that satisfies all rules and meta-rules of the system $\mathcal{WI}$, listed in figure 1 and figure 2.

Lemma 4 expresses that all rules derivable from $\mathcal{WI}$ are sound:

$$\varphi \vdash_X \psi \text{ implies } \varphi \models_X \psi$$

The converse holds as well. This can be proven by means of a *Henkin construction*.

The definition of $\vdash_X$ is extended to sets of formulas $(\Gamma, \Delta, \ldots)$, in order to get a standard deduction system. Compactness guarantees that this does not affect the consequence relation.

A Henkin construction involves maximal consistent sets of formulas. Consistency is by definition strong; maximality will be relative to a set of variables X. FORM(X) is the set of formulas with free variables in X.

Definition 8 *Γ is X-maximal consistent iff*
1. Γ is consistent: $\Gamma \not\vdash$ FALSE.
2. for any $\varphi \in$ FORM(X) such that $\varphi \notin \Gamma$: $\Gamma \cup \varphi$ inconsistent.

Every maximal consistent set satisfies a *truth lemma.*

Lemma 6 *Let Δ be X-maximal consistent. For all $\varphi, \psi \in$ FORM(X):*
1. $\varphi \in \Delta$ or $\neg\varphi \in \Delta$
2. $\varphi \vee \psi \in \Delta$ iff $\varphi \in \Delta$ or $\psi \in \Delta$
3. $\varphi \wedge \psi \in \Delta$ iff $\varphi \in \Delta$ and $\psi \in \Delta$

Proof: Straightforward: since $\varphi, \psi \in$ FORM(X) all rules reduce to their standard strong format. ∎
The existence of maximal X-consistent extensions will be the central issue in the completeness proof of $\mathcal{WI}$.

Theorem 7 *$\varphi \models_X \psi$ implies $\varphi \vdash_X \psi$.*

Proof: Outline; for further details see [6].
Suppose for some formulas φ_0, ψ_0 and some variables X_0 that $\varphi_0 \not\vdash_{X_0} \psi_0$.
We will construct a family of maximal consistent sets such that $\varphi_0 \in \Delta$ & $\psi_0 \notin \Delta$.
It can be assumed, without loss of generality, that $\alpha(\varphi_0) \subseteq X_0$.
The construction contains a number of steps:

1 Put $Y_0 := \alpha(\psi_0) \setminus X_0$,
so Y_0 is the set of existentially quantified variables in ψ_0.

2 Fix an enumeration $\{\theta_i\}_i$ of all equational solution sets $\theta : Y_0 \rightarrow$TERM(X_0),
where TERM(X_0) is the set of all object terms with variables in X_0.

3 Define $\Gamma := \varphi_0 \cup \{\neg\psi_0[\theta_i]\}_i$,
the extension of φ_0 with the set of all instantiations of $\neg\psi_0$.

Fact 1 Γ is consistent.
This can be proven formally in $\mathcal{WI}$, by deriving $\varphi_0 \vdash_{X_0} \psi_0$ from $\Gamma \vdash$ FALSE.

4 Embed Γ in a X_0-maximal consistent set Γ_*, by means of an enumeration $\{\xi_i\}_i$ of FORM(X_0), all formulas ξ with $\alpha(\xi) \subseteq X_0$.
1. $\Gamma_0 := \Gamma$
2. $\Gamma_{i+1} := \Gamma_i \cup \{\xi_i\}$, if this is consistent, $\Gamma_{i+1} := \Gamma_i$, otherwise.
3. $\Gamma_* := \bigcup_i \Gamma_i$

The restriction to FORM(X_0) entails that all implications are strong, so it is easy to prove the truth lemma (see lemma 6), restricted to formulas in FORM(X_0).

5 Define $\Delta^i := \Gamma_* \cup \{\theta_i\}$ for each θ_i in the enumeration.
Since already $\neg\psi_0[\theta_i] \in \Gamma_*$ this implies, by **generalization**, $\Delta^i \vdash \neg\psi_0$.

Fact 2 Every Δ^i is consistent.

6 Construct X_0Y_0-maximal extensions for each set Δ^i, as before, by means of an enumeration $\{\xi_j\}_j$ of FORM(X_0Y_0), all formulas ξ with $\alpha(\xi) \subseteq X_0Y_0$ (cf. 4).

The truth lemma is extended to FORM(X_0Y_0).
The resulting family of X_0Y_0-maximal sets is a counterexample to $\varphi_0 \models_{X_0} \psi_0$:

7 Define a model $\mathcal{M} =< \mathcal{D}, \mathcal{I} >$ with $\mathcal{D}$ the set of equivalence classes generated by Γ_* and $\mathcal{I}$ induced by Γ_*, as follows:

1. $\mathcal{D} := \{\underline{t} \mid \alpha(t) \subseteq X_0\}$, where $\underline{t} := \{s | s = t \in \Gamma_*\}$
2. $\mathcal{I}(c) := \underline{c}$, for all constants c.
3. $\mathcal{I}(P) := \{< \underline{t_1} \dots \underline{t_n} > \mid P(t_1 \dots t_n) \in \Gamma_*\}$, for all predicates P.
4. $\mathcal{I}(f) : \mathcal{D}^m \rightarrow \mathcal{D} := \mathcal{I}(f)(\underline{t_1} \dots \underline{t_n}) = \underline{f(t_1 \dots t_n)}$, for all functions f.

Obviously, Leibniz' Law is essential to make this a meaningful definition (see below).

8 The relevant assignments $h_i : \mathcal{A} \rightarrow \mathcal{D}$ are defined relative to the larger sets Δ^i_*:
$h_i(y) = \underline{t}$, if $y \in Y_0$, $y = t \in \theta_i$,
$h_i(x) = \underline{x}$, if $x \in X_0$.

This model is well-defined, that is, it satisfies the following properties.
Fact 3 For this model $< \mathcal{D}, \mathcal{I} >$ and these assignments h_i:

1. $\mathcal{M} \models P(t_1 \ldots t_n)[h_i]$ iff $P(t_1 \ldots t_n) \in \Delta^i_*$, $t_k \in \mathrm{TERM}(X_0 Y_0)$
2. $\mathcal{M} \models f(t_1 \ldots t_n) = s[h_i]$ iff $f(t_1 \ldots t_n) = s \in \Delta^i_*$
3. $\mathcal{M} \models \varphi[h_i]$ iff $\varphi \in \Delta^i_*$
4. $\mathcal{M} \models \xi[h_i]$ iff $\mathcal{M} \models \xi[h_j]$ for all $\xi \in \Gamma_*$, $\xi \in \mathrm{FORM}(X_0)$

9 $< \mathcal{D}, \mathcal{I} >$ is a well-defined countermodel for $\varphi_0 \models_{X_0} \psi_0$:
for all i: $\varphi_0 \in \Delta^i_*$, but $\psi_0 \notin \Delta^i_*$

As a consequence, $\mathcal{M} \models \varphi_0[h_i]$ and for all $h =_{X_0} h_i$, since $h = h_j$ for some j (!) $\mathcal{M} \not\models \psi_0[h]$, so $\mathcal{M} \not\models \varphi_0 \supset_{X_0} \psi_0$ and $\varphi_0 \not\models_{X_0} \psi_0$.

To summarize, we have constructed a countermodel to prove $\varphi_0 \not\models_{X_0} \psi_0$ for an arbitrary pair of formulas φ_0, ψ_0 and arbitrary set of variables X_0, on the assumption that $\varphi_0 \nvdash_{X_0} \psi_0$. Therefore, if $\varphi_0 \models_{X_0} \psi_0$, then $\varphi_0 \vdash_{X_0} \psi_0$; the system $\mathcal{WI}$ is complete.
■

5 Constraint Solving

The original motivation to introduce the notion of weak equivalence was to validate the **RL/1** symbolic constraint solving system (cf. [2], [3]). In this paper we do not discuss the technical details of this constraint solver, which can be found in [2]. We will illustrate the aims of the solver system by means of an example and give a formal specification based on weak equivalence.

5.1 An Example

Consider a large set of user-defined constraints, such as the arithmetic laws dealing with ages, wages and commissions on sales. These laws can be expressed in terms of universal constraints with several parameters and variables. Not all queries make use of these laws in the same manner: some queries may compute taxes over net prices, others require gross prices, but a curious buyer may want to reconstruct rates from gross and net prices. In all these queries the same equations are involved:

```
TaxConstraints:
gross-price = net-price + taxes,
taxes = rates * net-price,
etcetera.
```

For some applications gross prices are listed in the database, other products, that are mainly sold to business customers, are stored with their net prices: the known variables may vary with the query as well. Note that in a spreadsheet, such as EXCEL, all constraints are 'directed' in that new fields are computed out of given ones. Arithmetic equations as used in the relational database language **SQL** are directed as well: one may define new attributes as a function of given ones.

Informally, we would like to extend a database system with facilities to store relevant arithmetic laws in *constraints*. Queries concerning these constraints and the related database tables should be formulated in a user-friendly manner, for instance by allowing unrestricted selections in the **SQL** where-clause, either explicitly or by means of imported modules:

```
SELECT  net-price
  FROM  Orders, Clients, Rates
 WHERE  client-name = 'Me'
        net-price > 1000
        TaxConstraints(*)
```

(The natural join equations are suppressed; Rates contains the tax-rates that may vary per product.) The constraints in TaxConstraints may include equations that are irrelevant to the present query, such as boundaries on commissions per salesman, as a percentage of his salary. In principle, each tuple in the join Orders ⋈ Clients ⋈ Rates can be used to instantiate the constraints, testing for satisfiability to decide inclusion (: inconsistency leads to rejection). This type of evaluation is rather inefficient, however, since for all accepted tuples the constraint solver must be invoked to compute the net-price.

In a more efficient system, all unrestricted queries such as the one given above are preprocessed by a symbolic constraint solver, yielding a restricted query to be evaluated on the database:

```
CREATE VIEW  Query (net-price)
  AS SELECT  gross-price / (rates + 1)
       FROM  Orders, Clients, Rates
      WHERE  client-name = 'Me'
             gross-price > 1000 * rates + 1000
```

The necessity to employ a view, in order to identify net-price as the result of gross-price / (rates + 1), is of course an idiosyncrasy of the language **SQL**. However, it illustrates the difference between the *symbolic solution* and the actual answer. Note also the new restricted condition.

The symbolic answer is evaluated after the subsequent query:

```
SELECT  net-price
  FROM  Query
```

In this query only those tuples that will satisfy the constraints are selected to compute the net-price directly. The constraint solver has been invoked only once, to establish the symbolic solution and the remaining conditions, that is, the view definition.

More formally, the purpose of symbolic constraint solving is to determine whether or not a set of constraints is *separable* and if so, give an equivalent solution: the system should express the wanted variables W in terms of the known variables K, that have to satisfy some derived conditions. The input of a symbolic constraint solver is the unrestricted query $\Pi_X \sigma_\varphi(R)$. R is a database relation (or a view or join) to resolve known variables. The condition φ expresses all relevant knowledge; φ consists of equations and comparisons involving K, W and possibly some intermediate variables I, variables that are irrelevant to the present query, but that help to formulate the constraints in a user-friendly manner. The optimized output is a restricted database query of the format $\Pi_X(\kappa_\Phi(\sigma_\psi(R)))$, which can be evaluated directly on the database, without further intervention of the constraint solving system. The formal description and verification of the constraint solver can be formulated quite elegantly by means of the concept of weak equivalence.

5.2 Solvable Constraints

In general, a constraint solver is invoked to compute a relation $S(KW)$ from a relation $R(K)$ in accordance with the set of constraints φ. Let φ contain variables $\alpha(\varphi) = K \cup W \cup I$, that is, the disjoint union of known, wanted and intermediate variables. φ must be separated into a symbolic solution Φ and reduced condition ψ, but this will not be possible for all and arbitrary φ. The constraints φ are solvable in K and W if all variables in W can be expressed in terms of K; φ is reducible in K if the implicit condition φ poses on K can be expressed in terms of K alone.

- φ is **solvable** in K and W iff
  there exists a solution set Φ with $\alpha_H(\Phi) = W$ and $\alpha_T(\Phi) = K$, such that $\varphi \models_{KW} \Phi$.
- φ is **reducible** in K iff
  there exists a ψ with $\alpha(\psi) \subseteq K$, such that $\varphi \equiv_K \psi$.
- φ is **separable** in K and W iff
  there exists a solution set Φ with $\alpha_H(\Phi) = W$ and $\alpha_T(\Phi) = K$, and
  there exists a ψ with $\alpha(\psi) \subseteq K$, such that $\varphi \equiv_{KW} \Phi \wedge \psi$.

We say that φ is underdetermined on KW, if it is not solvable on K and W. (Note that in **CLP** the term *solvable* denotes satisfiability, $\varphi \equiv_\emptyset$ TRUE, that is, φ is not reducible to FALSE.)

A good optimization strategy first tries to establish whether or not a constraint φ is separable. To be separable φ must be solvable as well as reducible:

Theorem 8 *φ is separable in K and W iff*
φ is reducible in K and φ is solvable in K and W.

Proof: This can be proven formally in $\mathcal{WI}$, see figure 3 & 4.
■
Separability is a necessary requirement for efficient query evaluation. If the constraint φ is separable, then the constraint solver is only invoked once and the wanted variables can be computed by means of the symbolic solution Φ for all tuples that satisfy the reduced constraint ψ. In case φ is *not* separable, then only the more elaborate strategy remains of invoking the constraint solver for every tuple of known values, checking solvability for each individual tuple of known variables.

1	given: separable	$\varphi \equiv_{KW} \Phi \wedge \psi$
2	def: 1	$\varphi \models_{KW} \Phi \wedge \psi$
3	weakening: 2	$\varphi \models_{KW} \Phi$ (: solvable)
4	weakening: 2	$\varphi \models_{KW} \psi$
5	removal: 2	$\varphi \models_{K} \psi$
6	abstraction	$\psi \models_{K} \psi \wedge \Phi$
7	def:1	$\psi \wedge \Phi \models_{KW} \varphi$
8	transitivity	$\psi \models_{K} \varphi$
9	def: 5,8	$\varphi \equiv_{K} \psi$ (: reducible)

Figure 3: Separable implies solvable and reducible.

1	given: solvable	$\varphi \models_{KW} \Phi$
2	given: reducible	$\varphi \equiv_{K} \psi$
3	def: 2	$\varphi \models_{K} \psi$
4	irrelevance: 3	$\varphi \models_{KW} \psi$
5	union: 1,4	$\varphi \models_{KW} \psi \wedge \Phi$
6	def: 2	$\psi \models_{K} \varphi$
7	weakening: 6	$\psi \wedge \Phi \models_{K} \varphi$
8	augmentation: 1	$\varphi \models_{KW} \varphi \wedge \Phi$
9	transitivity: 7,8	$\psi \wedge \Phi \models_{K} \varphi \wedge \Phi$
10	substitution: 9	$\psi \wedge \Phi \models_{K} \varphi[\Phi]$
11	irrelevance: 10	$\psi \wedge \Phi \models_{KW} \varphi[\Phi]$
12	weakening	$\psi \wedge \Phi \models_{KW} \Phi$
13	union: 11,12	$\psi \wedge \Phi \models_{KW} \Phi \wedge \varphi[\Phi]$
14	generalization	$\Phi \wedge \varphi[\Phi] \models_{KW} \varphi \wedge \Phi$
15	transitivity: 13,14	$\psi \wedge \Phi \models_{KW} \varphi \wedge \Phi$
16	transitivity: 15	$\psi \wedge \Phi \models_{KW} \varphi$
17	def: 5, 16	$\varphi \equiv_{KW} \psi \wedge \Phi$ (: separable)

Figure 4: Solvable and reducible implies separable.

5.3 Symbolic Solutions

We would like to verify that the symbolic solution yields all and only correct answers, at least for separable constraints. Hence we must compare two *types* of constraint solvers, say $T1$ and $T2$. $T1$ corresponds with tuple-wise evaluation, and $T2$ with the more efficient evaluation strategy that employs symbolic solutions.

Definition 9 *Two types of constraint solvers.*

T1 *input:* φ, W; *output:* ψ, Φ.	**T2** *input:* φ, K, W; *output:* ψ, Φ.
1. $\varphi \equiv_W \psi \wedge \Phi$	1. $\varphi \equiv_{KW} \psi \wedge \Phi$
2. $W \subseteq \alpha(\varphi)$	2. $KW \subseteq \alpha(\varphi)$, $K \cap W = \emptyset$
3. $\psi \equiv$ TRUE *or* $\psi \equiv$ FALSE	3. $\alpha(\psi) \subseteq K$
4. $\alpha_T(\Phi) = \emptyset$	4. $\alpha_T(\Phi) \subseteq K$
5. $\alpha_H(\Phi) = W$	5. $\alpha_H(\Phi) = W$

Both solvers are partial in the sense that the output constraint ψ and the solution Φ are only generated if φ is not underdetermined on KW.

For a type $T1$ solver the solution Φ is a *tuple* which contains values for all wanted variables W. The constraint φ is solvable if $\psi \equiv$ TRUE, and inconsistent if $\psi \equiv$ FALSE. In the latter case a dummy Φ can be constructed to satisfy the specification. The type $T1$ solver is invoked for each tuple r in the relation R, with input W and $\varphi[r]$, the result of the substitution r on φ. Its output is a new tuple s (: over the attributes W) and an error-indicator ψ. In case $\psi \equiv$ TRUE, then $\varphi[r] \equiv_W s$, and the tuple $s \sqcup r$ is added to the answer relation S. If $\psi \equiv$ FALSE, then r is incompatible with φ, that is, $\varphi[r]$ is inconsistent, and no tuple is added to S.

A type $T2$ solver generalizes a $T1$ solver by the introduction of known variables. It generates a *symbolic* solution Φ and a *symbolic* solvability condition ψ. In case $\psi \equiv$ FALSE the query is rejected. Otherwise, solvability can be checked by simple evaluation of ψ for all tuples r in R. Those tuples r that pass the test can be extended to tuples $r \sqcup s$ on KW by calculating $s := \Phi[r]$. Therefore, the requested answer relation S can be computed with the following restricted database expression:

$$S \;:=\; \kappa_\Phi(\sigma_\psi(R)) \;=\; \{t \in S \mid \exists r \in R:\; t = r \sqcup s \;\&\; r \models \psi \;\&\; s = \Phi[r]\}$$

This is formalized in the following lemma.

Lemma 9 *If φ is separable, $\varphi \equiv_{KW} \psi \wedge \Phi$, and $\alpha(R) = K$, then for all $r \in R$ and all $s \in \mathcal{D}^W$: $\varphi[r \sqcup s] \equiv_\emptyset$ TRUE iff $r \models \psi$ & $s = \Phi[r]$.*

Proof:
1. Suppose $\varphi[r \sqcup s] \equiv_\emptyset$ TRUE.
Then $r \sqcup s \models_{KW} \varphi$, so, since $\varphi \equiv_{KW} \psi \wedge \Phi$, $r \sqcup s \models_{KW} \psi \wedge \Phi$, and, since $\alpha(\psi) \subseteq \alpha(r)$, $r \models_{KW} \psi$, that is, $\psi[r] \equiv$ TRUE. Moreover, from $r \sqcup s \models_{KW} \Phi$, by **instantiation**, $(r \sqcup s)[r] \models_{KW} \Phi[r]$, that is, $s \models_{KW} \Phi[r]$. On account of the scopes involved this means that $s = \Phi[r]$.
2. Suppose that $\psi[r] \equiv_K$ TRUE and $s = \Phi[r]$.
Then $r \models \psi$ and $s \models \Phi[r]$, so, by **mixed union** $r \sqcup s \models \Phi[r] \wedge r \wedge \psi$, and, by **generalization**

$r \sqcup s \models \Phi \wedge \psi$. Since $\varphi \equiv_{KW} \psi \wedge \Phi$, this implies $r \sqcup s \models_{KW} \varphi$, hence $\varphi[r \sqcup s] \equiv_{\emptyset}$ TRUE.
■

As soon as W is determined by φ in terms of K the $T2$ solver must be preferred to the direct strategy that invokes $T1$ for each individual tuple. If the problem is underdetermined, however, it may turn out that some tuples are capable of being solved, as in the following example.

Example 7 $T2$: *input:* $a * x + b * y = c$, $K = \{a, b, c\}$, $W = \{x\}$; *output: ? ? ?*
For r *such that* $r(b) = 0$ *the* $T1$ *solver gets input* $a * x + b * y = c[r]$, $\{x\}$, *yielding the answer* TRUE, $\{x = (c/a)[r]\}$. *Indeed, if* $r(a) * x = r(c)$, *then* $x = r(c)/r(a)$.

The $T1$ strategy is more complete in the sense that it can deal with inseparable constraints that happen to be separable after substitution for all tuples in R. Nevertheless, it may be assumed that the $T1$ solver is not invoked for all tuples of R on account of the general underdeterminedness of the constraint. Hence it is not unfair to change the constraint of example 7 for both solvers into the determined query $(a * x + b * y = c) \ \wedge \ (a = 0 \vee b = 0)$.

The $T2$ constraint solver that was described in this section has been implemented in the **RL/1** optimization strategy developed at the University of Amsterdam by S. van Denneheuvel (see [2]).

The $T1$ constraint solver corresponds with algorithms like *Gauss elimination* to solve sets of linear equations φ, where $W = \alpha(\varphi)$, and with the binary REDUCE procedure from *Mathematica.*

The system **CLP** ([5]) has a different strategy. A successful derivation (based on unification, as in PROLOG) expands a goal φ relative to the program P to yield a set of answer constraints. This corresponds with evaluating $\varphi[\theta]$ for a tuple θ (: atomic data) and does not require constraint solving. Then a type $T1$ solver is invoked to solve the remaining variables W from $\varphi[\theta]$ for each tuple θ. However, symbolic answers can be computed as well, by invocation of a type $T2$ solver with input φ, $K = \alpha(\varphi)$, $W = \emptyset$ and output ψ, $\Phi = \emptyset$ such that $\varphi \equiv_{\alpha(\varphi)} \psi$.

6 Query Normalization

Query optimization techniques appreciate standardized input, but, unfortunately, there is no general normal form for terms in the relational algebra. There are unconditional rewrite rules for a large fragment of the algebra though, leaving but a small set of cumbersome queries. In this section we will illustrate how weak equivalence can be used to normalize relational terms, yielding a normal form for almost all terms.

Relational terms are constructed from relation names by means of algebraic operators Π (: projection), σ (: selection), $\cup$ (: union), $\bowtie$ (: join), $\setminus$ (: relational difference) and [ ] (: renaming). Every term T is interpreted as a finite relation over its scope $\alpha(T)$, both defined by the usual induction (cf. [8] or [6]).

Definition 10 *Two relational terms* R *and* S *are equivalent,* $R \equiv S$, *if for every database* $< \mathcal{D}, \mathcal{A}, \mathcal{I} >$: $\mathcal{I}(R) = \mathcal{I}(S)$.

Example 8 $\sigma_\varphi\sigma_\psi(R) \equiv \sigma_{\varphi\wedge\psi}(R)$,
$\sigma_\varphi(R) \setminus \sigma_\psi(S) \equiv \sigma_{\varphi\wedge\neg\psi}(R) \cup \sigma_\varphi(R \setminus S)$.

Renaming is essentially a syntactic operation, that is, $R[B/A]$ is a notational variant of R, renaming all attributes A in the constituent relations in R to B and actually performing the renaming in the projection- and selection sets. For basic relations $R[B/A]$ is a view over R, the same table with a new heading. Hence we may 'perform' all occurrences of renaming instantaneously on the terms themselves, leaving only 5 operations to be normalized.

Many relational equivalences are straightforward and well-known: cascades of projections and selections, set equivalences, union distribution, miscellaneous selection equivalences and the like (see e.g. [8] or [10]). Here only the projection equivalences will be mentioned.

Lemma 10 *$R \bowtie \Pi_X(S) \equiv \Pi_{X\cup\alpha(R)}(R \bowtie S[Y/Z])$,
where $Y := (\alpha(R) \cap \alpha(S)) \setminus X$ and $Z \cap \alpha(T) = \emptyset$, for all basic relations T.*

Proof: See [6], pg 52 ff.
■
The proof of this lemma is not very complicated, but it is tricky and a little messy. However, it can be formulated as a weak equivalence by means of a view Q for the projection subterm $\Pi_X(S)$ and a solution set Φ with 'new' variables as tails for the appropriate renaming of clashing variables:

Lemma 11 *If $Q :- \Pi_X(S)$, then $R \bowtie Q \equiv_{X\cup\alpha(R)} R \bowtie S[\Phi]$,
where $\alpha_H(\Phi) := (\alpha(R) \cap \alpha(S)) \setminus X$ and $\alpha_T(\Phi) \cap \alpha(T) = \emptyset$, for all basic relations T.*

Proof: Since $\alpha(Q) = X$, $Q \equiv_X S$. If $\alpha(R) \cap \alpha(S) \subseteq X$, then $R \bowtie Q \equiv_{X\cup\alpha(R)} R \bowtie S$. No need to give a proof; this is the **augmentation** rule! To avoid the condition, define a renaming solution set $\Phi := \{A_1 = B_1, \ldots\}$, with $\alpha_H(\Phi) = (\alpha(R) \cap \alpha(S)) \setminus X$ and $\alpha_T(\Phi) \cap \alpha(T) = \emptyset$, for all basic relations T. From **abstraction** we infer $Q \equiv_X S$ iff $Q \equiv_X S[\Phi]$. Moreover, $\alpha(R) \cap \alpha(S[\Phi]) \subseteq X$ by definition of Φ.
■

Example 9 *If $R(ABC)$ and $S(BCD)$, then: $R \bowtie \Pi_{CD}(S) \equiv \Pi_{ABCD}(R \bowtie S[E/B])$, in other words, if $Q \equiv_{CD} S$, then $R \bowtie Q \equiv_{ABCD} R \bowtie S[B = E]$.*

By a similar argument, we can pull projection over selection:

Lemma 12 *If $Q :- \Pi_X(R)$, then $\sigma_\varphi(Q) \equiv_{X\cup\alpha(\varphi)} \sigma_\varphi(R[\Phi])$,
where $\alpha_H(\Phi) := (\alpha(R) \cap \alpha(\varphi)) \setminus X$ and $\alpha_T(\Phi) \cap \alpha(T) = \emptyset$, for all basic relations T.*

The union operator interchanges freely with projection, though the resulting union need not be of compatible terms.

Lemma 13 *If $Q :- \Pi_X(R)$, then $Q \cup S \equiv_{X\cup\alpha(R)} R[\Phi] \cup S$
where $\alpha_H(\Phi) := (\alpha(R) \cap \alpha(S)) \setminus X$ and $\alpha_T(\Phi) \cap \alpha(T) = \emptyset$, for all basic relations T.*

The difference operator interchanges with projection on the positive side:

Constraint type	**Example format**
functional / primary key	$R(xyz), R(xy'z') \models y = y'$
inclusion / foreign key	$R(xy) \models_x S(xz)$
embedded multivalued	$R(xyzw), R(xy'z'w') \models_{xyzy'z'} R(xy'zw'')$
lossless join	$R(xyzw) \equiv_{xyzw} R(xyz'w'), R(xy'zw'), R(x'yzw)$

Figure 5: Examples of integrity constraints

Lemma 14 *If* $Q : -\ \Pi_X(R)$, *then* $Q \setminus S \equiv_{X \cup \alpha(R)} R[\Phi] \setminus S$, *where* $\alpha_H(\Phi) := (\alpha(R) \cap \alpha(S)) \setminus X$ *and* $\alpha_T(\Phi) \cap \alpha(T) = \emptyset$, *for all basic relations* T.

Projection cannot be pulled over the negative side of a difference operator. This corresponds with the restriction on **contraposition**: from $Q \equiv_X S$ we cannot infer $\neg Q \equiv_X \neg S$, except when $\alpha(S) = X$ (in which case the projection would be trivial). Hence $R \setminus \Pi_X(S)$ cannot be expressed by a single equivalence statement (see [6]).

All positive occurrences of projection can be pulled to the outside, so if φ is a formula to express the projection-free subterm R, then $Q\ :-\ \Pi_X(R)$ is expressed by $Q(\mathbf{x}) \equiv_{\mathbf{x}} \varphi$. In this manner any relational term can be translated into a finite set of weak equivalences, one for every occurrence of the projection operator (cf. [8], pg 154 ff.). In particular, if all occurrences of projections in a relational term T are 'positive', then a single equivalence statement suffices to express T.

7 Integrity Constraints

There is yet another possible application of the notion of weak equivalence, namely to describe integrity constraints. Consider a set of functional- and inclusion dependencies, or a set of primary - and foreign key dependencies or even a set of embedded multivalued dependencies. If these constraints must be translated into predicate formulas one needs universal and existential quantifiers:

functional dependency	$\forall x, y, y' :\ R(xy) \wedge R(xy') \supset y = y'$
inclusion dependency	$\forall x, y :\ R(xy) \supset \exists z :\ S(xz)$

Note that identical variables are used to match attributes in R with corresponding attributes in S. To avoid quantification one can use weak implications, see figure 5.

The Armstrong axioms for functional dependencies and similar rules for other types of dependencies translate into valid implications that are derivable from $\mathcal{WI}$. To give an example of a more complicated rule, consider the mixed rule for functional and inclusion dependencies:

$$R[XY] \subseteq S[XY],\ R[XZ] \subseteq S[XZ],\ S :\ X \to Y \ \Rightarrow\ R[XYZ] \subseteq S[XYZ]$$

This rule translates into the following weak implication rule:

$$\left.\begin{array}{l} Rxyzu \models_{xy} Sxyzw, \\ Rxyzu \models_{xz} Sxyzw, \\ Sxyzw \wedge Sxy'z'w' \models y = y' \end{array}\right\} \Rightarrow Rxyzu \models_{xyz} Sxyzw$$

Given this notation for the mixed rule we can employ the (meta-) rules of weak implications to prove its correctness. The resulting proof is rigorous and simple and compares favourably with the usual informal argument in terms of R and S tuples.

Generally speaking, the formulation of integrity constraints by means of weak implication statements is hardly preferable to the standard notation. Still, it is very convenient for formal verification arguments, and can be used as a tool to integrate integrity with *query optimization* and *views.*

For instance, a *foreign key dependency* $\text{FK}(R, X, Q)$ may link the relation R to a previously defined view Q with *primary key* $\text{PK}(S) = X$. The expanded definition of Q contains a set of basic relations. One would like to infer the induced constraints on these basic relations from the constraints on Q.

Any view can be expressed by a finite set of weak equivalences, a single one if we ignore projection on the negative side of a difference (see § 6). Once views and integrity constraints are formulated in the same framework, it is possible to integrate the implication problems. We will only give a simple example to explain the basic idea: weak implications can be applied to constraints on views in order to induce constraints on basic relations:

Lemma 15 *Let* $\text{FK}(R, X, Q)$ *and* $Q :- S \bowtie T \cup S \bowtie U$.
Suppose X *is such that both* $X \subseteq \alpha(S)$ *and* $\alpha(T \bowtie U) \cap X = \emptyset$. *Then* $\text{FK}(R, X, S)$.

Proof: In terms of weak equivalences:

given : *foreign key*	$R(XY) \models_X Q(XZ)$
given : *view def.*	$Q(XZ) \equiv_{XZ} S(XYW) \wedge (T(YWZ) \vee U(YZ))$
weakening	$Q(XZ) \models_{XZ} S(XYW)$
removal	$Q(XZ) \models_X S(XYW)$
transitivity	$R(XY) \models_X S(XYW)$

This proves that $R[X] \subseteq S[X]$. By a similar argument one can show that $\text{PK}(S) = X$. ■
The integration works both ways: it can also be employed for query optimization purposes. In the following example the number of relations under a projection is reduced on account of the integrity constraint.

Lemma 16 *If* $X = \alpha(R) \cap \alpha(S)$ *and* $\text{FK}(R, X, S)$, *then* $R \bowtie S \equiv_X R$.

Proof: Let $Y = \alpha(R) \setminus X$ and $Z = \alpha(S) \setminus X$, then we need to prove:

$$R(XY) \models_X S(XZ) \;\Rightarrow\; R(XY), S(XZ) \equiv_X R(XY).$$

Straightforward, cf. [2] (§ Join optimization).
■
The examples in this section are not very profound, but they are only simple illustrations of a fundamental possibility: the application of the notion of weak equivalence on integrity constraints, views and query optimization. To do so in a systematic way remains as a future task.

8 Conclusion

We have given a sound and complete set of deduction rules for weak implications. This notion of implication generalizes unquantified predicate logic with a single level of $\forall\exists$ quantification. This reduced class of predicate formulas is adequate for a large variety of subjects, ranging from formal specifications for constraint solving systems to implication problems in database theory. Since weak equivalence combines notational transparency with formal elegance, it offers a convenient semantic tool for knowledge representation.

References

[1] W.F. Clocksin & C.S. Mellish. *Programming in Prolog.* Springer-Verlag, 1981.

[2] S. van Denneheuvel. *Constraint solving on database systems. Design and implementation of the rule language* **RL/1**. Thesis. University of Amsterdam, 1991.

[3] S. v Denneheuvel & K.L. Kwast. *Weak equivalence for constraint solving.* In: Proceedings of IJCAI'91: Int. Joint Conference on A.I. Sydney. Morgan Kaufmann, 1991.

[4] S. v Denneheuvel, G.R. Renardel de Lavalette, E. Spaan & K.L. Kwast. *Query optimization using rewrite rules.* In: R. Book (ed.), Proceedings of RTA'91: Int. Conference on Rewriting Techniques and Applications. Como, LNCS488, 1991.

[5] J. Jaffar & S. Michaylov. *Methodology and Implementation of a CLP System.* In: J-L. Lassez (ed.), Proceedings of the 4th Int. Conference on Logic Programming, MIT Press, 1987.

[6] K.L. Kwast. *Unknown values in the relational database system.* Thesis. University of Amsterdam, 1992.

[7] J.H. Siekmann. *Unification Theory.* In: Journal of Symbolic Computation, Vol. 7, 1989.

[8] J.D. Ullman. *Principles of Data and Knowledge-Base Systems, Vol. I & II.* Computer Science Press, 1989.

[9] S. Wolfram. *Mathematica, a System for Doing Math by Computer.* Addison-Wesley, 1988.

[10] H.Z. Yang & P.A. Larson. *Query Transformation for PSJ-queries.* In: Proceedings of the 13th VLDB: Int. Conference on Very Large Databases, 1987.

The ILLC Prepublication Series

ML-91-05 A.S. Troelstra	History of Constructivism in the Twentieth Century
ML-91-06 Inge Bethke	Finite Type Structures within Combinatory Algebras
ML-91-07 Yde Venema	Modal Derivation Rules
ML-91-08 Inge Bethke	Going Stable in Graph Models
ML-91-09 V.Yu. Shavrukov	A Note on the Diagonalizable Algebras of PA and ZF
ML-91-10 Maarten de Rijke, Yde Venema	Sahlqvist's Theorem for Boolean Algebras with Operators
ML-91-11 Rineke Verbrugge	Feasible Interpretability
ML-91-12 Johan van Benthem	Modal Frame Classes, revisited

Computation and Complexity Theory

CT-91-01 Ming Li, Paul M.B. Vitányi	Kolmogorov Complexity Arguments in Combinatorics
CT-91-02 Ming Li, John Tromp, Paul M.B. Vitányi	How to Share Concurrent Wait-Free Variables
CT-91-03 Ming Li, Paul M.B. Vitányi	Average Case Complexity under the Universal Distribution Equals Worst Case Complexity
CT-91-04 Sieger van Denneheuvel, Karen Kwast	Weak Equivalence
CT-91-05 Sieger van Denneheuvel, Karen Kwast	Weak Equivalence for Constraint Sets
CT-91-06 Edith Spaan	Census Techniques on Relativized Space Classes
CT-91-07 Karen L. Kwast	The Incomplete Database
CT-91-08 Kees Doets	Levationis Laus
CT-91-09 Ming Li, Paul M.B. Vitányi	Combinatorial Properties of Finite Sequences with high Kolmogorov Complexity
CT-91-10 John Tromp, Paul Vitányi	A Randomized Algorithm for Two-Process Wait-Free Test-and-Set
CT-91-11 Lane A. Hemachandra, Edith Spaan	Quasi-Injective Reductions
CT-91-12 Krzysztof R. Apt, Dino Pedreschi	Reasoning about Termination of Prolog Programs

Computational Linguistics

CL-91-01 J.C. Scholtes	Kohonen Feature Maps in Natural Language Processing
CL-91-02 J.C. Scholtes	Neural Nets and their Relevance for Information Retrieval
CL-91-03 Hub Prüst, Remko Scha, Martin van den Berg	A Formal Discourse Grammar tackling Verb Phrase Anaphora

Other Prepublications

X-91-01 Alexander Chagrov, Michael Zakharyaschev	The Disjunction Property of Intermediate Propositional Logics
X-91-02 Alexander Chagrov, Michael Zakharyaschev	On the Undecidability of the Disjunction Property of Intermediate Propositional Logics
X-91-03 V. Yu. Shavrukov	Subalgebras of Diagonalizable Algebras of Theories containing Arithmetic
X-91-04 K.N. Ignatiev	Partial Conservativity and Modal Logics
X-91-05 Johan van Benthem	Temporal Logic
X-91-06	Annual Report 1990
X-91-07 A.S. Troelstra	Lectures on Linear Logic, Errata and Supplement
X-91-08 Giorgie Dzhaparidze	Logic of Tolerance
X-91-09 L.D. Beklemishev	On Bimodal Provability Logics for Π_1-axiomatized Extensions of Arithmetical Theories
X-91-10 Michiel van Lambalgen	Independence, Randomness and the Axiom of Choice
X-91-11 Michael Zakharyaschev	Canonical Formulas for K4. Part I: Basic Results
X-91-12 Herman Hendriks	Flexibele Categoriale Syntaxis en Semantiek: de proefschriften van Frans Zwarts en Michael Moortgat
X-91-13 Max I. Kanovich	The Multiplicative Fragment of Linear Logic is NP-Complete
X-91-14 Max I. Kanovich	The Horn Fragment of Linear Logic is NP-Complete
X-91-15 V. Yu. Shavrukov	Subalgebras of Diagonalizable Algebras of Theories containing Arithmetic, revised version
X-91-16 V.G. Kanovei	Undecidable Hypotheses in Edward Nelson's Internal Set Theory
X-91-17 Michiel van Lambalgen	Independence, Randomness and the Axiom of Choice, Revised Version
X-91-18 Giovanna Cepparello	New Semantics for Predicate Modal Logic: an Analysis from a standard point of view
X-91-19 Papers presented at the Provability Interpretability Arithmetic Conference, 24-31 Aug. 1991, Dept. of Phil., Utrecht University	
1992	Annual Report 1991

Logic, Semantics and Philosophy of Langauge

LP-92-01 Víctor Sánchez Valencia	Lambek Grammar: an Information-based Categorial Grammar
LP-92-02 Patrick Blackburn	Modal Logic and Attribute Value Structures
LP-92-03 Szabolcs Mikulás	The Completeness of the Lambek Calculus with respect to Relational Semantics
LP-92-04 Paul Dekker	An Update Semantics for Dynamic Predicate Logic
LP-92-05 David I. Beaver	The Kinematics of Presupposition
LP-92-06 Patrick Blackburn, Edith Spaan	A Modal Perspective on the Computational Complexity of Attribute Value Grammar
LP-92-07 Jeroen Groenendijk, Martin Stokhof	A Note on Interrogatives and Adverbs of Quantification
LP-92-08 Maarten de Rijke	A System of Dynamic Modal Logic

Mathematical Logic and Foundations

ML-92-01 A.S. Troelstra	Comparing the theory of Representations and Constructive Mathematics
ML-92-02 Dmitrij P. Skvortsov, Valentin B. Shehtman	Maximal Kripke-type Semantics for Modal and Superintuitionistic Predicate Logics
ML-92-03 Zoran Marković	On the Structure of Kripke Models of Heyting Arithmetic
ML-92-04 Dimiter Vakarelov	A Modal Theory of Arrows, Arrow Logics I

Compution and Complexity Theory

CT-92-01 Erik de Haas, Peter van Emde Boas	Object Oriented Application Flow Graphs and their Semantics
CT-92-02 Karen L. Kwast, Sieger van Denneheuvel	Weak Equivalence: Theory and Applications

Other prepublications

X-92-01 Heinrich Wansing	The Logic of Information Structures
X-92-02 Konstantin N. Ignatiev	The Closed Fragment of Dzhaparidze's Polymodal Logic and the Logic of Σ_1-conservativity
X-92-03 Willem Groeneveld	Dynamic Semantics and Circular Propositions, revised version