Institute for Language, Logic and Information

WEAK EQUIVALENCE

Sieger van Denneheuvel
Karen Kwast

ITLI Prepublication Series
for Computation and Complexity Theory CT-91-04

University of Amsterdam

Instituut voor Taal, Logica en Informatie
Institute for Language, Logic and Information

Faculteit der Wiskunde en Informatica
(Department of Mathematics and Computer Science)
Plantage Muidergracht 24
1018TV Amsterdam

Faculteit der Wijsbegeerte
(Department of Philosophy)
Nieuwe Doelenstraat 15
1012CP Amsterdam

WEAK EQUIVALENCE

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

ITLI Prepublication Series
for Computation and Complexity Theory
ISSN 0924-8374

Received March 1991

Weak equivalence

Sieger van Denneheuvel & Karen Kwast
Department of Mathematics and Computer Science
Plantage Muidergracht 24, 1018 TV, University of Amsterdam

Abstract

In this paper we describe a generalization of equivalence between constraint sets, called *weak equivalence*. This equivalence relation takes into account that not all variables have the same function in a constraint set and therefore distinguishes between *restriction variables* and *intermediate variables*. We explore the properties of weak equivalence and its underlying notion of weak implication with an axiomatic approach and we present a complete set of axioms for weak implication. As an application of the axiomatization we derive a general tool for constraint set simplification. Two constraint solving strategies are described in terms of weak equivalence. Furthermore we briefly compare the notion of weak equivalence with equivalence of substitution with respect to a set of variables as used in unification theory. The paper ends with a detailed proof of the soundness of the axioms inferred for weak implication.

1 Introduction

Recently, systems have emerged that allow declarative constraint processing (see for example Mathematica [WOLF88], Bertrand [LELER88], **CLP($\Re$)** [JAF87] and **RL/1** [DEN90]) whereas more traditional systems required user intervention to direct the solver towards a desired solution. From a general point of view, a typical input for these systems consists of a set of *constraints* and a set of *restriction variables* which are "interesting" variables as far as the solver is concerned. The remaining variables in the set of constraints are *intermediate variables.* For constraint solving, restriction variables can be further divided in two mutually disjunct sets namely *known variables* and *wanted variables.*

Given the above input the objective of the constraint solver is to express all wanted variables symbolically in terms of known variables (i.e. without supplying actual values for the known variables). In such a symbolic solution only restriction variables are allowed and as a consequence all intermediate variables must be eliminated from the input constraint set. As a simple example consider the input constraint set $\{x = y + 2, y = z + 2\}$ where z is known and x is wanted, so x and z together are the restriction variables. A constraint solver set to solve this problem will return $\{x = z + 4\}$ as the solution, thereby eliminating the intermediate variable y. For such a solver output, we would like to say that it is in some way equivalent with the input constraint set. Unfortunately the standard definition for equivalence among constraint sets would classify the example input and output as inequivalent, since the output set does not impose any restriction to the intermediate variable y.

Therefore we define a more general equivalence, called *weak equivalence* which distinguishes between restriction variables and intermediate variables. In this paper we explore the properties of weak equivalence and the underlying notion of weak implication with an axiomatic approach. Weak equivalence can be used to express the unification problem (see [SIEK89]).

Our primary motivation for investigation of weak equivalence lies in its role in the integration of relational databases and constraint solving. This integration is one of the aims of the declarative rule language family **RL**. In the **RL** languages, knowledge can be represented in three different types of rules: *tabular rules*, *clauses* and *constraints.* Corresponding to these types of rules there are three areas of technology that support that style of knowledge processing in isolation: database systems, logic programming systems, and spreadsheets. A main goal of **RL** is to integrate these three technologies in one system. Query processing should be executed with the help of an existing relational database system; knowledge and queries expressed in **RL** are preprocessed by a constraint solver, and compiled into a relational algebra query language so that large amounts of data can be processed effectively. In the current prototype implementation **RL/1** we do not focus on recursion as is done in the NAIL! system [ULL89] and the **LDL** deductive database [NAQV89], but rather on the integration of a subsystem solving (numeric) constraints and a relational database system; the architecture for such an integrated system has been presented elsewhere [DEN88].

The **RL** language is a declarative representation of knowledge. This means that the user who wants to express knowledge in **RL** rules should not have to worry about control issues in the representation, but only needs to specify what he believes is true in the represented domain. The representation of the rules should also be *auditable* in the sense that the written text can be inspected by a non-technician in order to convince himself that the rules in the system indeed represent those in the world outside.

In the **RL** language design a program consists of modules, each module describing a system of relations. Relations are described using expressions originating from the worlds of relational databases, logic programming and equational logic. Atomic relations can be combined using operators originating from these worlds, providing maximal freedom and full conceptual transparency to the user.

A main objective of **RL** is sharing of both rules and data. It is often desirable to incorporate into a relational system a representation of common knowledge shared by different users, so that multiple developments of similar programs can be avoided. At this point the relational database becomes a *knowledge base*. In **RL**, representation of common knowledge is facilitated by modularization. Modules enable the user to manage and organize a large collection of rules in a structured way. For more complete information on **RL** we refer to [VEMD86a] and [VEMD86b]. For information on the evaluation of clausal rules on relational databases we refer to [VEMD86c].

2 Weak equivalence and implication

First we introduce some definitions and notations. Variable names are chosen from non capitals $(x, y, z, \ldots)$ and sets of variables from capitals $(X, Y, Z, \ldots)$. We denote the set of all variables by $\mathbf{V}$. Constants are denoted by c and functions by f. For use in the database application domain we include, for instance, binary operators in the set $\{+, -, *, /\}$. Terms $(s, t, \ldots)$ are constructed inductively in the usual way from constants, variables and functions. *Constraints* $(a, b, \ldots)$ include expressions of the form t_1 op t_2 with $op \in \{=, >, <, \neq, \leq, \geq\}$ and two special constraints **false** and **true**. Sets of constraints are represented as A, B, C or D. To denote the cardinality of a set I we use the notation $\|I\|$.

The output of a constraint solver will be a solution set, which expresses the wanted variables in terms of known variables.

Definition 1 *A solution set is a finite set* $\{x_1 = t_1, \ldots, x_n = t_n\}$*, satisfying:*
1. $\|\{x_1, \ldots, x_n\}\| = n$, *(the variables are distinct)*
2. $\{x_1, \ldots, x_n\} \cap var(\{t_1, \ldots, t_n\}) = \emptyset$

A *tuple* is a solution set of the form $\{x_1 = c_1, \ldots, x_n = c_n\}$ where $c_1, \ldots, c_n$ are constants. Solution sets are denoted by Φ, Ψ and Θ and tuples by ϕ, ψ and θ.

Example 1 *(Solution sets and tuples)*
$\{name =$ *'bob'*, $age = 55$, $dep =$ *'toy'*$\}$ *(a tuple)*
$\{x = 1, y = 2, z = 3\}$ *(a tuple)*
$\{x = u + 2, y = v + 2\}$ *(a solution set)*
$\{x = u + 2, y = x + 2\}$ *(not a solution set)*
$\{x = u + 2, x = v + 2\}$ *(not a solution set)*

Definition 2 *(Head and tail variables of a solution set)*
1. $hv(\{x_1 = t_1, \ldots, x_n = t_n\}) := \{x_1, \ldots, x_n\}$
2. $tv(\{x_1 = t_1, \ldots, x_n = t_n\}) := var(t_1, \ldots, t_n)$

Definition 3 *(The restriction of* Φ *to* X *)*
$\Phi[X] := \{x = t \mid x \in X, x = t \in \Phi\}$

Definition 4 *(Restricted equality on solution sets)*
$\Phi =_X \Psi \quad iff \quad \Phi[X] = \Psi[X]$

Solution sets $\Phi = \{x_1 = t_1, \ldots, x_n = t_n\}$ can be interpreted as substitutions $[x_1 := t_1, \ldots, x_n := t_n]$ which can be applied to (sets of) constraints. So we have an operation *apply*:

Definition 5 $\Phi(A) := A[x_1 := t_1, \ldots, x_n := t_n]$ *with* $\Phi = \{x_1 = t_1, \ldots, x_n = t_n\}$

Example 2 *(Substitution)*
$\{x = u + 2\}(\{x = u + 1\}) = \{u + 2 = u + 1\}$
$\{x = u + 2, y = v + 2\}(\{x > y\}) = \{u + 2 > v + 2\}$

A tuple ϕ satisfies a set of constraints A, denoted $\phi \models A$, if and only if ϕ satisfies all constraints in A. A set of constraints A implies a set of constraints B, denoted $A \models B$, if ϕ satisfies B whenever ϕ satisfies A, for all tuples ϕ.

Definition 6 *(Strong satisfaction, implication and equivalence)*
1. $\phi \models A \quad iff \quad \forall a \in A\ (\phi \models a)$
2. $A \models B \quad iff \quad \forall \phi\ (\phi \models A \Rightarrow \phi \models B)$
3. $A \equiv B \quad iff \quad A \models B\ \&\ B \models A \quad iff \quad \forall \phi\ (\phi \models A \Leftrightarrow \phi \models B)$

To construct weak equivalence the above definitions are subscripted with the restriction variables X. Strong satisfaction is changed to weak satisfaction by existentially quantifying the variables not in X (i.e. the intermediate variables):

Definition 7 *(Weak satisfaction, implication and equivalence)*
1. $\phi \models_X A \quad iff \quad \exists \psi\ (\psi \models A\ \&\ \phi[X] = \psi[X])$
2. $A \models_X B \quad iff \quad \forall \phi\ (\phi \models A \Rightarrow \phi \models_X B)$
3. $A \equiv_X B \quad iff \quad A \models_X B\ \&\ B \models_X A$

Example 3 *(Implication)*
$\{x = 1\} \models_{\{x\}} \{y = 1\}$
$\{x = y\} \models_{\{x\}} \{x = y + 1\}$
$\{x = y\} \models_{\{x\}} \{x \neq y\}$
$\{x = 1\} \not\models_{\{x\}} \{x = 0\}$
$\{y = 1\} \not\models_{\{x\}} \{x = 1\}$

Example 4 *(Equivalence)*
$\{x = 1\} \not\equiv_{\{x,y\}} \{y = 1\}$
$\{x = 1\} \not\equiv_{\{x\}} \{y = 1\}$
$\{x = 1\} \equiv_{\emptyset} \{x = 0\}$
false $\not\equiv_{\emptyset}$ **true**
$\{name = \text{'bob'}\} \equiv_{\{city\}} \{name = \text{'cathy'}\}$
$\{x = y\} \equiv_{\{x\}} \{x = y + 1\}$

3 Axioms for equivalence and implication

In this section we enumerate some axioms that hold for implication and equivalence. We start with properties of strong equivalence that are maintained for weak equivalence.

Proposition 8 *(The equivalence relation $\equiv_X$)*
1. $A \equiv_X A$
2. $A \equiv_X B \Rightarrow B \equiv_X A$
3. $A \equiv_X B$ & $B \equiv_X C \Rightarrow A \equiv_X C$

The following axioms are specifically concerned with the restriction variables of weak implication.

Proposition 9 *(Addition and removal of restriction variables)*
1. $A \models_X B \Rightarrow A \models_{X \cup Y} B$ *if* $var(B) \cap Y \subset X$
2. $A \equiv_X B \Rightarrow A \equiv_{X \cup Y} B$ *if* $var(A \cup B) \cap Y \subset X$
3. $A \models_{X \cup Y} B \Rightarrow A \models_X B$
4. $A \models B \Rightarrow A \models_X B$

Proposition 10 *(Axioms for weak implication)*
1. $A \cup B \models_X A$
2. $A \models_X B$ & $B \models_X C \Rightarrow A \models_X C$
3. $A \models_X B$ & $A \models_X C \Rightarrow A \models_X B \cup C$ *if* $var(B) \cap var(C) \subset X$
4. $A \models_X B \Rightarrow A \cup C \models_X B \cup C$ *if* $var(B) \cap var(C) \subset X$
5. $A \models_X B \cup C \Rightarrow A \models_X B$
6. $A \models_X B$ & $B \cup C \models_X D \Rightarrow A \cup C \models_X D$ *if* $var(B) \cap var(C) \subset X$
7. $A \models_X B \cup C$ & $B \models_X D \Rightarrow A \models_X C \cup D$ *if* $var(C) \cap var(D) \subset X$
8. $A \models_X B$ & $C \models_X D \Rightarrow A \cup C \models_X B \cup D$ *if* $var(B) \cap var(D) \subset X$

The *strong* versions of axioms (1), (2) and (3) are known in Functional Dependency Theory as *weakening*, *transitivity* and *union* respectively (cf. [ULL88] and [VARDI88]). From these axioms all other axioms (4)-(8) can be derived. The axioms of Proposition 10 are truly more general than their strong counterparts, since from the axioms (1)-(8) strong unconditional axioms can be derived. For instance $A \models B$ & $A \models C \Rightarrow A \models B \cup C$ can be derived from (3) by assigning to X the set of all variables $\mathbf{V}$ in which case the condition becomes trivially true.

Inconsistency is normally defined as strong equivalence with **false** and tautology as strong equivalence with **true**. It turns out that weak and strong inconsistency are actually equivalent. However, contrary to inconsistency, weak tautology is truly weaker than strong tautology:

Proposition 11 *(Inconsistency and tautology)*
1. $A \equiv_X \textbf{false} \Leftrightarrow A \equiv \textbf{false}$
2. $A \equiv \textbf{true} \Rightarrow A \equiv_X \textbf{true}$
3. $A \equiv_X \textbf{true} \not\Rightarrow A \equiv \textbf{true}$

It is easy to construct an example for (3). Let $A = \{x < x + y, y > 0\}$ then it holds that $A \equiv_{\{x\}}$ **true**. On the other hand $A \not\equiv$ **true** because $\{x < x + y, y > 0\} \equiv \{y > 0\}$.

The equivalence relation $\equiv_\emptyset$ has only two equivalence classes, **true** and **false**:

Name	Axiom
True	$A \models \textbf{true}$
False	$\textbf{false} \models A$
Weakening	$A \cup B \models A$
Transitivity	$A \models B \ \& \ B \models C \Rightarrow A \models C$
Union	$A \models B \ \& \ A \models C \Rightarrow A \models B \cup C$
Substitution	$A \cup \Phi \models \Phi(A)$
Generalization	$\Phi(A) \cup \Phi \models A$
Instantiation	$A \models B \Rightarrow \Phi(A) \models \Phi(B)$

Figure 1: Axiom system $\mathcal{AS}$ for strong implication

Proposition 12 $A \equiv_\emptyset$ **true** *or* $A \equiv_\emptyset$ **false.**

Note that $A \equiv_\emptyset$ **true** does *not* imply that $var(A) = \emptyset$ or that $A \equiv$ **true**. On the contrary, $A \equiv_\emptyset$ **true** expresses that A is consistent, and $A \equiv_\emptyset$ **false** means that A is inconsistent. For application of weak equivalence to constraint solving, the following substitution axioms have to be added:

Proposition 13 *(Substitution axioms)*
1. $A \cup \Phi \equiv_X \Phi(A) \cup \Phi$
2. $\Phi(A) \models_X A \cup \Phi$ *if* $hv(\Phi) \cap X = \emptyset$
3. $A \models B \Rightarrow \Phi(A) \models \Phi(B)$
4. $A \models_X B \Rightarrow \Phi(A) \models_X \Phi(B)$ *if* $var(\Phi) \cap var(B) \subset X$

Axiom (2) allows you to *add* solutions for intermediate variables. Axiom (2) is never applicable for strong implication.

4 Completeness for strong and weak implication

A natural question that arises in the context of an axiomatic approach is whether a complete axiomatization is feasible. Fortunately both for weak and strong implication, a completeness result can be derived. Completeness was proven with respect to *primitive* constraints, that is constraints of the format '$x = y$' or '$x = c$'.

Theorem 14 *For primitive constraints the axioms in $\mathcal{AS}$ and $\mathcal{AW}$ are sound and complete*

The completeness proof of this theorem will be presented in [DEN91]. Section 7 contains the soundness proof.

Using the axiom system $\mathcal{AW}$ a proposition can be established that specifically exploits weak equivalence for constraint elimination. The idea is that a subset B of a set of constraints $A \cup B$ is redundant with respect to weak equivalence if there exists a solution set Φ such that $\Phi(B)$, the effect of substituting Φ on B, is equivalent with **true**:

Theorem 15 $A \cup B \equiv_X A$ *if there exists some* Φ *such that*
1. $\Phi(B) \equiv$ **true**
2. $hv(\Phi) \cap X = \emptyset$
3. $hv(\Phi) \cap var(A) = \emptyset$

Name	Axiom	Condition
True	$A \models_X$ **true**	
False	**false** $\models_X A$	
Weakening	$A \cup B \models_X A$	
Transitivity	$A \models_X B$ & $B \models_X C \Rightarrow A \models_X C$	
Union	$A \models_X B$ & $A \models_X C \Rightarrow A \models_X B \cup C$	$var(B) \cap var(C) \subset X$
Substitution	$A \cup \Phi \models_X \Phi(A) \cup \Phi$	
Generalization	$\Phi(A) \cup \Phi \models_X A \cup \Phi$	
Abstraction	$\Phi(A) \models_X A \cup \Phi$	$hv(\Phi) \cap X = \emptyset$
Removal	$A \models_{X \cup Y} B \Rightarrow A \models_X B$	
Irrelevance	$A \models_X B \Rightarrow A \models_{X \cup Y} B$	$var(B) \cap Y \subset X$

Figure 2: Axiom system $\mathcal{AW}$ for weak implication

Proof: i) $A \cup B \models_X A$: Weakening.
ii) $A \models_X A \cup B$:
$A \models_X A \Rightarrow(3)\ A \models_X \Phi(A) \Rightarrow A \models_X \Phi(A) \cup \textbf{true} \Rightarrow(1)\ A \models_X \Phi(A) \cup \Phi(B)$
$\Rightarrow A \models_X \Phi(A \cup B) \Rightarrow$(2, Abstraction) $A \models_X A \cup B \cup \Phi \Rightarrow$(Weakening) $A \models_X A \cup B$
■

Example 5 *(Constraint elimination)*

$\{x = y + 2, y = z + 2\} \equiv_{\{x,y\}} \{x = y + 2\}$ *with* $\Phi = \{z = y - 2\}$
$\{u > x, x > v, u > v\} \equiv_{\{u,v\}} \{u > v\}$ *with* $\Phi = \{x = (u + v)/2\}$
$\{v = u + 3, x > u\} \equiv_{\{u,v\}} \{v = u + 3\}$ *with* $\Phi = \{x = u + 1\}$
$\{v = u + 3, x > u\} \not\equiv_{\{u,v,x\}} \{v = u + 3\}$ *by (2) x may not be substituted*
$\{x + y = 0, y > 0\} \not\equiv_{\{x\}} \{x + y = 0\}$ *by (3) $\Phi = \{y = 1\}$ is illegal*

5 Application of weak equivalence

Solving a set of constraints is essentially a two stage process:

- Determine the solvability of the constraints.
- If found solvable then solve the constraints.

In this section we describe strategies *S1* and *S2* to perform this task when the constraints are to be solved repeatedly for a large collection of tuples ψ of a database table $R(K)$ with attributes K. The aim of constraint solving is to extend each tuple $\psi \in R(K)$ with a tuple θ such that $hv(\theta) = W$. The set of extended tuples constitutes a new relation with attributes $K \cup W$ called *answer relation S* in the sequel.
With the strategies *S1* and *S2* two solver types *T1* and *T2* are associated:

Definition 16 *(The T1 solver and T2 solver)*

T1$(A, W, B, \Phi) :=$
1. $A \equiv_W B \cup \Phi$
2. $W \subset var(A)$
3. $B =$ **true** *or* $B =$ **false**
4. $tv(\Phi) = \emptyset$
5. $hv(\Phi) = W$

T2$(A, K, W, B, \Phi) :=$
1. $A \equiv_{K \cup W} B \cup \Phi$
2. $K \subset var(A), W \subset var(A), K \cap W = \emptyset$
3. $var(B) \subset K$
4. $tv(\Phi) \subset K$
5. $hv(\Phi) = W$

For type *T1* the set of constraints A is to be solved for the set of wanted variables W. For an input (A, W) a *T1* solver generates a solution set Φ and a condition set B satisfying (*T1*.1). The *T1* solver is a partial function in the sense that the output sets B and Φ are only generated if A is not underdetermined on W. The solution set Φ is a tuple which contains values for all wanted variables W ((*T1*.4) and (*T1*.5)). The constraint set A is either solvable if B = **true** or inconsistent if B = **false** (*T1*.3). In the latter case an arbitrary solution Φ can be constructed in accordance to (*T1*.4) and (*T1*.5) without violating (*T1*.1).

A type *T2* solver generalizes a *T1* solver by the introduction of known variables K. For an input (A, K, W) a *T2* solver generates a *symbolic* solution Φ and a *symbolic* solvability condition B. If $B \equiv$ **false** then the original constraint set A is inconsistent. Analogously to *T1* also *T2* is a partial function. By (*T2*.4) and (*T2*.5) a solution for all wanted variables W can be calculated directly for a selected tuple $\psi \in R(K)$ by evaluation of $\psi(\Phi)$. For a particular tuple ψ the expression $\psi(B)$ either yields **true** or **false** (see (*T2*.3) above). So the actual checking of solvability of the constraints in A is reduced into simple evaluation of an expression. The *T2* solver eliminates all intermediate variables I $(:= var(A) - K - W)$ from both B and Φ.

In general, the computation of the answer relation S can be achieved using two strategies:

1. In strategy *S1* the solver is invoked for each tuple $\psi \in R(K)$ as follows:

   $$T1(\psi(A), W, B, \Phi)$$

   The solver returns either B = **true** or B = **false**. In the former case the tuple Φ contains *values* for all wanted variables W and the tuple $\psi \cup \Phi$ is added to the answer relation S. In latter case (i.e. the constraints in A are inconsistent for the selected tuple ψ) no tuple is added to S.

2. In strategy *S2* the solver is invoked as follows:

   $$T2(A, K, W, B, \Phi)$$

   The solver returns the symbolic solution Φ and the condition set B that represents the solvability of the original constraint set A. Inside the database the requested answer relation S can be efficiently computed with the following relational algebra operations:

   $$S = \kappa(\sigma(R(K), B), \Phi)$$

In the above expression first the relation $T = \sigma(R(K), B) := \{\psi \mid \psi \in R(K), \psi \models B\}$ is computed. Next the tuples in T are extended by the calculate operator which computes the relation $S = \kappa(T, \Phi) := \{\psi \cup \theta \mid \psi \in T, \theta = \psi(\Phi)\}$.

The validity of strategy *S2* is verified by means of the following theorem:

Theorem 17
If $T2(A, K, W, B, \Phi)$ *then forall* ψ *with* $hv(\psi) = K$ *and forall* θ *with* $hv(\theta) = W$*:*

$$\theta(\psi(A)) \equiv_\emptyset \textbf{true} \Leftrightarrow \psi(B) \equiv \textbf{true} \ \& \ \theta = \psi(\Phi)$$

A serious drawback of strategy *S1* is that the co-operation between solver and relational database is based on tuple transfer and the solver is invoked separately for each tuple. In this strategy special care must be taken to optimize this interface between database and constraint solver (cf. **CLP**($\Re$) [JAF87]). Moreover this approach prohibits the use of existing relational database systems, since these are normally not equipped with constraint solving capabilities.

In strategy *S2* the answer relation S was obtained by translation to a relational query that can be executed directly on the relational database. For the translation the solver is invoked only *once* for all tuples $\psi \in R(K)$.

A *T2* constraint solver has been implemented as a sequence of transformations of the original constraint set A. The in- and outputs of each of these transformations were formally specified and subsequently the transformations themselves were implemented. Using the weak axioms presented in this paper we proved that after each of the intermediate transformations, weak equivalence with respect to the original set A is kept as an invariant [DEN91]. This implemented constraint solver uses, among others, the next proposition to throw away unnecessary solutions:

Proposition 18 $A \cup \{x = t\} \equiv_{K \cup W} A$ *if*
1. $x \notin K \cup W$
2. $x \notin var(A)$

Proof: Let $\Phi = \{x = t\}$ then $\Phi(\{x = t\}) = \{t = t\} \equiv \textbf{true}$ so (1) from Proposition 15 is satisfied. The conditions from Proposition 15 reduce directly to the conditions of this proposition.

■

The above proposition serves as a theoretical justification for elimination of solutions $x = t$ for variables x that are intermediate and hence do not occur in the restriction set $K \cup W$. In this case the condition (2) ensures that the variable x is indeed eliminated from the constraint set A.

6 Comparison with unification theory

One of the referees of an earlier version of this paper drew our attention to the close parallel between weak equivalence and equivalence of substitutions with respect to a set of variables, as used in unification theory.

In general a unifier for a set of equations A is a substitution σ such that $\sigma(A) \equiv \textbf{true}$. Here we restrict ourselves to independent sets of solutions Φ, that is, substitutions that are separated away from their head variables (see definition 1; cf. [SIEK89]). The unification problem for a set of terms $\mathcal{T}$ over an equational theory E is to find the most general unifiers for a pair (s, t). In other words, find Φ such that $E \Rightarrow \Phi \models s = t$. The relation between unification and constraint solving now becomes clear. We claim that *T2* describes the unification problem for unitary theories, since if

$$T2(A, K, W, \textbf{true}, \Phi)$$

with $K \cup W = var(A)$, then $A \equiv \Phi$, so $\Phi \models A$ and Φ is a unifier. Moreover, since $A \models \Phi$, Φ is certainly the most general unifier. There is one major distinction between unification and constraint solving and that is the assumption of the *freeness axioms*. To make sure that the constraint $f(x,b) = f(a,y)$ implies its unifier $\Phi = \{x = a, y = b\}$ we need a freeness axiom $f(x_1, x_2) = f(y_1, y_2) \models x_1 = y_1, x_2 = y_2$.

Unfortunately, if f is also known to be commutative, or to distributes over a second function g, the freeness axiom must be reformulated (adding the axiom $f(x,y) = f(y,x)$ would reduce the domain to a singleton !). Hence in general the "unification axioms" of a function can be rather complicated.

In constraint solving, we are not looking for a unifier, but for the solved form of the set of equations or constraints. Nevertheless, as soon as *T2* is satisfied it yields a most general unifier:

Example 6 $T2(2^x * 3^y = 2^a * 3^b, \{a,b\}, \{x,y\}, \mathbf{true}, \{x = a, y = b\})$.
In this case $\Phi = \{x = a, y = b\}$ *is the most general unifier of* $A = \{2^x * 3^y = 2^a * 3^b\}$.
$T2(x + a * b = y * b + x, \{x,a,b\}, \{y\}, \mathbf{true}, y = a)$ *with m.g.u.* $\Phi = \{y = a\}$.

Needless to say that *T2* can only be used on the so called *unitary* unification problems, and that the specification itself gives no indication as how to construct Φ. The specification *T2* can be extended to yield a disjunctive constraint set B or, to cover *finitary* unifiers, a set of sets with corresponding solutions, so:

$$A \equiv_{K \cup W} (B_1 \cup \Phi_1) \vee \ldots \vee (B_n \cup \Phi_n)$$

Since B need not be **true** or **false**, even if W is determined by A, we can also deal with *partial* unification, reducing not only the number of equations to be unified, but also the number of variables in the remaining constraints.

As an example consider the following sequence of examples:

Example 7

- $T2(\{x + a * b = y * b + x, 2^x * 3^b = 2^y * 3^a\}, \{x,a,b\}, \{y\}, \{2^x * 3^b = 2^a * 3^a\}, \{y = a\})$
- $T2(\{2^x * 3^b = 2^a * 3^a\}, \{a,b\}, \{x\}\{3^b = 3^a\}, \{x = a\})$
- $T2(\{3^b = 3^a\}, \{a\}, \{b\}\{\mathbf{true}\}, \{b = a\})$
- *To combine the above three steps we derive:*

$$T2(\{x + a * b = y * b + x, 2^x * 3^b = 2^y * 3^a\}, \{a\}, \{b,x,y\}, \mathbf{true}, \{y = a, x = a, b = a\})$$

The main motivation to use weak equivalence, however, is to remove intermediate variables. As far as term unification is concerned, there are no intermediate variables, but in constraint solving there are. For instance for

$$A = \{a - b = m, c - d = n, a * x = c, b * x = d, n + c > b * x\}$$

we derive $T2(A, \{n,m\}, \{x\}, \{n > 0\}, \{x = n/m\})$. As a result of Theorem 17 we can now be sure that the answer relation S can be constructed from $R(n,m)$ with the following relational operations:

$$\kappa(\sigma(R(n,m), n > 0), x = n/m)$$

7 Proofs of the axioms of $\mathcal{AW}$

In this section we proof the soundness of the axioms in $\mathcal{AW}$.

Proposition 19 .... [*weakening*]
$A \cup B \models_X A$

Proof:
$\forall \phi \; (\phi \models A \cup B \Rightarrow \phi \models_X A)$
■

Proposition 20 .... [*transitivity*]
$A \models_X B \;\&\; B \models_X C \Rightarrow A \models_X C$

Proof:
$A \models_X B \;\&\; B \models_X C$
$\Rightarrow \forall \phi \; (\phi \models A \Rightarrow \phi \models_X B) \;\;\&\;\; \forall \psi \; (\psi \models B \Rightarrow \psi \models_X C)$
$\Rightarrow \forall \phi \; (\phi \models A \Rightarrow \exists \psi \; (\psi \models B \;\&\; \phi =_X \psi))$
$\&\;\; \forall \psi \; (\psi \models B \Rightarrow \exists \theta \; (\theta \models C \;\&\; \psi =_X \theta))$
$\Rightarrow \forall \phi \; (\phi \models A \Rightarrow \exists \theta \; (\theta \models C \;\&\; \phi =_X \theta))$
$\Rightarrow \forall \phi \; (\phi \models A \Rightarrow \phi \models_X C)$
$\Rightarrow A \models_X C$
■

Proposition 21 .... [*union*]
$A \models_X B \;\&\; A \models_X C \Rightarrow A \models_X B \cup C \quad \textit{if} \quad var(B) \cap var(C) \subset X$

Proof:
$A \models_X B \;\&\; A \models_X C$
$\Rightarrow \forall \theta \; (\theta \models A \Rightarrow \exists \psi \; (\psi \models B \;\&\; \theta =_X \psi))$
$\&\;\; \forall \theta \; (\theta \models A \Rightarrow \exists \psi' \; (\psi' \models C \;\&\; \theta =_X \psi'))$
$\Rightarrow \forall \theta \; (\theta \models A \Rightarrow \exists \phi \; (\phi \models B \cup C \;\&\; \theta =_X \phi))$ .... $(*)$
$\Rightarrow A \models_X B \cup C$
To verify step $(*)$ we construct an appropriate ϕ from ψ and ψ':

$$\phi = \psi[var(B)] \cup \psi'[var(C)]$$

Clearly the tuple ϕ satisfies both B and C. Furthermore it can be easily established that ϕ is well defined and that $\theta =_X \phi$, as required.
■

Proposition 22 .... [*removal*]
1. $\phi \models_{X \cup Y} A \Rightarrow \phi \models_X A$
2. $A \models_{X \cup Y} B \Rightarrow A \models_X B$

Proof:
1. $\phi \models_{X \cup Y} A \Rightarrow \exists \psi \; (\psi \models A \;\&\; \phi =_{X \cup Y} \psi)$
$\Rightarrow \exists \psi \; (\psi \models A \;\&\; \phi =_X \psi)$
$\Rightarrow \phi \models_X A$
2. $A \models_{X \cup Y} B$

$\Rightarrow \forall\phi\ (\phi \models A \Rightarrow \phi \models_{X \cup Y} B)$
$\Rightarrow \forall\phi\ (\phi \models A \Rightarrow \phi \models_X B)$ .... (1)
$\Rightarrow A \models\!\!\!=_X B$
■

Proposition 23 .... [*irrelevance*]
1. $\phi \models_X A \Rightarrow \phi \models_{X \cup Y} A \quad$ *if* $\quad var(A) \cap Y \subset X$
2. $A \models\!\!\!=_X B \Rightarrow A \models\!\!\!=_{X \cup Y} B \quad$ *if* $\quad var(B) \cap Y \subset X$

Proof:
1. $\phi \models_X A \Rightarrow \exists\psi\ (\psi \models A\ \&\ \phi =_X \psi))$
$\Rightarrow \exists\psi\ (\psi \models A\ \&\ \phi =_{X \cup Y} \psi))$
$\Rightarrow \phi \models_{X \cup Y} A$
2. Follows from (1). ■

Proposition 24 .... [*substitution*]
1. $\theta \models A \cup \Phi \Rightarrow \theta \models \mathbf{\Phi}(A)$
2. $A \cup \Phi \models\!\!\!= \mathbf{\Phi}(A)$
3. $A \cup \Phi \models\!\!\!=_X \mathbf{\Phi}(A) \cup \Phi$

Proof:
1. Assume that $\Phi = \{x_1 = t_1, \ldots, x_n = t_n\}$.
$\theta \models A \cup \Phi$
$\Leftrightarrow \theta \models A(x_1, \ldots, x_n) \cup \{x_1 = t_1, \ldots, x_n = t_n\}$
$\Rightarrow \theta \models A(t_1, \ldots, t_n) \cup \{x_1 = t_1, \ldots, x_n = t_n\}$
$\Rightarrow \theta \models A(t_1, \ldots, t_n)$
$\Rightarrow \theta \models \mathbf{\Phi}(A)$
2. Follows from (1).
3. Follows from (2).
■

Proposition 25 .... [*generalization*]
1. $\theta \models \mathbf{\Phi}(A) \cup \Phi \Rightarrow \theta \models A \cup \Phi$
2. $\mathbf{\Phi}(A) \cup \Phi \models\!\!\!= A \cup \Phi$
3. $\mathbf{\Phi}(A) \cup \Phi \models\!\!\!=_X A \cup \Phi$

Proof:
1. Assume that $\Phi = \{x_1 = t_1, \ldots, x_n = t_n\}$.
$\theta \models \mathbf{\Phi}(A) \cup \Phi$
$\Rightarrow \theta \models A(t_1, \ldots, t_n) \cup \Phi$
$\Rightarrow \theta \models A(x_1, \ldots, x_n) \cup \Phi$
$\Leftrightarrow \theta \models A \cup \Phi$
2. Follows from (1).
3. Follows from (2).
■

Proposition 26 .... [*abstraction*]
1. $\theta \models \mathbf{\Phi}(A) \Rightarrow \theta \models_X A \cup \Phi \quad$ *if* $\quad hv(\Phi) \cap X = \emptyset$
2. $\mathbf{\Phi}(A) \models\!\!\!=_X A \cup \Phi \quad$ *if* $\quad hv(\Phi) \cap X = \emptyset$

Proof:

1. Assume that $\Phi = \{x_1 = t_1, \ldots, x_n = t_n\}$.

$\theta \models \Phi(A)$

$\Rightarrow \theta \models A(t_1, \ldots, t_n)$

$\Rightarrow \exists \psi \ (\psi \models A(x_1, \ldots, x_n) \cup \{x_1 = t_n, \ldots, x_n = t_n\} \ \& \ \theta =_X \psi)$ ... (using $hv(\Phi) \cap X = \emptyset$)

$\Leftrightarrow \exists \psi \ (\psi \models A \cup \Phi \ \& \ \theta =_X \psi)$

$\Leftrightarrow \theta \models_X A \cup \Phi$

2. Follows from (1).

■

References

[DEN88] van Denneheuvel, S. & van Emde Boas, P., *Constraint solving for databases,* Proc. of NAIC **1**, 173-184, (1988).

[DEN90] van Denneheuvel, S. & van Emde Boas, P., *The rule language RL/1,* A. M. Tjoa & R. Wagner (eds.), Database and Expert Systems Applications, Proc. of the International Conference, Vienna, Austria, Aug 1990, Springer Verlag Wien, 381-387, (1990).

[DEN91] van Denneheuvel, S. & Kwast, K. L., *Weak equivalence for constraint solving,* Preprint, Institute for Language, Logic and Information, University of Amsterdam, to appear (1991).

[JAF87] Jaffar, J. & Michaylov, S., *Methodology and Implementation of a CLP System,* Logic Programming, Proc. of the Fourth Int. Conf., Ed. Jean-Louis Lassez, 196-218, (1987).

[LELER88] Leler, Wm., *Constraint Programming Languages (Their Specification and Generation),* Addison-Wesly Series in Computer Science, Addison-Wesly, (1988).

[NAQV89] Naqvi, S. & Tsur, S., *A Logical Language for Data and Knowledge bases,* Computer Science Press, (1989).

[SIEK89] Siekmann, J. H., *Unification Theory,* J. of Symb. Comp. 7, 207-274, (1989)

[ULL88] Ullman, J.D., *Principles of Data and Knowledge - Base Systems,* Volume I, Computer Science Press, (1988).

[ULL89] Ullman, J.D., *Principles of Data and Knowledge - Base Systems,* Volume II, Computer Science Press, (1989).

[VARDI88] Vardi, M. Y., *Fundamentals of Dependency Theory,* In: Trends in Theoretical Computer Science, Egon Borger (ed.), Computer Science Press, 171-225, (1988).

[VEMD86a] van Emde Boas, P., *RL, a Language for Enhanced Rule Bases Database Processing,* Working Document, Rep IBM Research, RJ 4869 (51299), (1986).

[VEMD86b] van Emde Boas, P., *A semantical model for the integration and modularization of rules,* Proceedings MFCS **12**, Bratislava, Springer Lecture Notes in Computer Science **233**, 78-92, (1986).

[VEMD86c] van Emde Boas, H. & van Emde Boas, P., *Storing and Evaluating Horn-Clause Rules in a Relational Database,* IBM J. Res. Develop. **30** (1), 80-92, (1986).

[WOLF88] Wolfram, S., *Mathematica, A System for Doing Mathematics by Computer,* Addison-Wesly, (1988).

The ITLI 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

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

Computation and Complexity Theory

CT-91-01 Ming Li, Paul M.B. Vitanyi	Kolmogorov Complexity Arguments in Combinatorics
CT-91-02 Ming Li, John Tromp, Paul M.B. Vitanyi	How to Share Concurrent Wait-Free Variables
CT-91-03 Ming Li, Paul M.B. Vitanyi	Average Case Complexity under the Universal Distribution Equals Worst Case Complexity
CT-91-04 Sieger van Denneheuvel, Karen Kwast	Weak Equivalence

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 Diagonizable Algebras of Theories containing Arithmetic
X-91-04 K.N. Ignatiev	Partial Conservativity and Modal Logics
X-91-05 Johan van Benthem	Temporal Logic

The ITLI 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

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

Computation and Complexity Theory

CT-91-01 Ming Li, Paul M.B. Vitanyi	Kolmogorov Complexity Arguments in Combinatorics
CT-91-02 Ming Li, John Tromp, Paul M.B. Vitanyi	How to Share Concurrent Wait-Free Variables
CT-91-03 Ming Li, Paul M.B. Vitanyi	Average Case Complexity under the Universal Distribution Equals Worst Case Complexity
CT-91-04 Sieger van Denneheuvel, Karen Kwast	Weak Equivalence

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 Diagonizable 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

The ITLI Prepublication Series

1986

86-01 The Institute of Language, Logic and Information
86-02 Peter van Emde Boas — A Semantical Model for Integration and Modularization of Rules
86-03 Johan van Benthem — Categorial Grammar and Lambda Calculus
86-04 Reinhard Muskens — A Relational Formulation of the Theory of Types
86-05 Kenneth A. Bowen, Dick de Jongh — Some Complete Logics for Branched Time, Part I Well-founded Time, Forward looking Operators
86-06 Johan van Benthem — Logical Syntax

1987

87-01 Jeroen Groenendijk, Martin Stokhof — Type shifting Rules and the Semantics of Interrogatives
87-02 Renate Bartsch — Frame Representations and Discourse Representations
87-03 Jan Willem Klop, Roel de Vrijer — Unique Normal Forms for Lambda Calculus with Surjective Pairing
87-04 Johan van Benthem — Polyadic quantifiers
87-05 Víctor Sánchez Valencia — Traditional Logicians and de Morgan's Example
87-06 Eleonore Oversteegen — Temporal Adverbials in the Two Track Theory of Time
87-07 Johan van Benthem — Categorial Grammar and Type Theory
87-08 Renate Bartsch — The Construction of Properties under Perspectives
87-09 Herman Hendriks — Type Change in Semantics: The Scope of Quantification and Coordination

1988

Logic, Semantics and Philosophy of Language:

LP-88-01 Michiel van Lambalgen — Algorithmic Information Theory
LP-88-02 Yde Venema — Expressiveness and Completeness of an Interval Tense Logic
LP-88-03 — Year Report 1987
LP-88-04 Reinhard Muskens — Going partial in Montague Grammar
LP-88-05 Johan van Benthem — Logical Constants across Varying Types
LP-88-06 Johan van Benthem — Semantic Parallels in Natural Language and Computation
LP-88-07 Renate Bartsch — Tenses, Aspects, and their Scopes in Discourse
LP-88-08 Jeroen Groenendijk, Martin Stokhof — Context and Information in Dynamic Semantics
LP-88-09 Theo M.V. Janssen — A mathematical model for the CAT framework of Eurotra
LP-88-10 Anneke Kleppe — A Blissymbolics Translation Program

Mathematical Logic and Foundations:

ML-88-01 Jaap van Oosten — Lifschitz' Realizabiility
ML-88-02 M.D.G. Swaen — The Arithmetical Fragment of Martin Löf's Type Theories with weak Σ-elimination
ML-88-03 Dick de Jongh, Frank Veltman — Provability Logics for Relative Interpretability
ML-88-04 A.S. Troelstra — On the Early History of Intuitionistic Logic
ML-88-05 A.S. Troelstra — Remarks on Intuitionism and the Philosophy of Mathematics

Computation and Complexity Theory:

CT-88-01 Ming Li, Paul M.B.Vitanyi — Two Decades of Applied Kolmogorov Complexity
CT-88-02 Michiel H.M. Smid — General Lower Bounds for the Partitioning of Range Trees
CT-88-03 Michiel H.M. Smid, Mark H. Overmars, Leen Torenvliet, Peter van Emde Boas — Maintaining Multiple Representations of Dynamic Data Structures
CT-88-04 Dick de Jongh, Lex Hendriks, Gerard R. Renardel de Lavalette — Computations in Fragments of Intuitionistic Propositional Logic
CT-88-05 Peter van Emde Boas — Machine Models and Simulations (revised version)
CT-88-06 Michiel H.M. Smid — A Data Structure for the Union-find Problem having good Single-Operation Complexity
CT-88-07 Johan van Benthem — Time, Logic and Computation
CT-88-08 Michiel H.M. Smid, Mark H. Overmars, Leen Torenvliet, Peter van Emde Boas — Multiple Representations of Dynamic Data Structures
CT-88-09 Theo M.V. Janssen — Towards a Universal Parsing Algorithm for Functional Grammar
CT-88-10 Edith Spaan, Leen Torenvliet, Peter van Emde Boas — Nondeterminism, Fairness and a Fundamental Analogy
CT-88-11 Sieger van Denneheuvel, Peter van Emde Boas — Towards implementing RL

Other prepublications:

X-88-01 Marc Jumelet — On Solovay's Completeness Theorem

1989

Logic, Semantics and Philosophy of Language:

LP-89-01 Johan van Benthem — The Fine-Structure of Categorial Semantics
LP-89-02 Jeroen Groenendijk, Martin Stokhof — Dynamic Predicate Logic, towards a compositional, non-representational semantics of discourse
LP-89-03 Yde Venema — Two-dimensional Modal Logics for Relation Algebras and Temporal Logic of Intervals
LP-89-04 Johan van Benthem — Language in Action
LP-89-05 Johan van Benthem — Modal Logic as a Theory of Information
LP-89-06 Andreja Prijatelj — Intensional Lambek Calculi: Theory and Application
LP-89-07 Heinrich Wansing — The Adequacy Problem for Sequential Propositional Logic
LP-89-08 Víctor Sánchez Valencia — Peirce's Propositional Logic: From Algebra to Graphs
LP-89-09 Zhisheng Huang — Dependency of Belief in Distributed Systems

Mathematical Logic and Foundations:

ML-89-01 Dick de Jongh, Albert Visser — Explicit Fixed Points for Interpretability Logic
ML-89-02 Roel de Vrijer — Extending the Lambda Calculus with Surjective Pairing is conservative
ML-89-03 Dick de Jongh, Franco Montagna — Rosser Orderings and Free Variables
ML-89-04 Dick de Jongh, Marc Jumelet, Franco Montagna — On the Proof of Solovay's Theorem
ML-89-05 Rineke Verbrugge — Σ-completeness and Bounded Arithmetic
ML-89-06 Michiel van Lambalgen — The Axiomatization of Randomness
ML-89-07 Dirk Roorda — Elementary Inductive Definitions in HA: from Strictly Positive towards Monotone
ML-89-08 Dirk Roorda — Investigations into Classical Linear Logic
ML-89-09 Alessandra Carbone — Provable Fixed points in $I\Delta_0+\Omega_1$

Computation and Complexity Theory:

CT-89-01 Michiel H.M. Smid — Dynamic Deferred Data Structures
CT-89-02 Peter van Emde Boas — Machine Models and Simulations
CT-89-03 Ming Li, Herman Neuféglise, Leen Torenvliet, Peter van Emde Boas — On Space Efficient Simulations
CT-89-04 Harry Buhrman, Leen Torenvliet — A Comparison of Reductions on Nondeterministic Space
CT-89-05 Pieter H. Hartel, Michiel H.M. Smid, Leen Torenvliet, Willem G. Vree — A Parallel Functional Implementation of Range Queries
CT-89-06 H.W. Lenstra, Jr. — Finding Isomorphisms between Finite Fields
CT-89-07 Ming Li, Paul M.B. Vitanyi — A Theory of Learning Simple Concepts under Simple Distributions and Average Case Complexity for the Universal Distribution (Prel. Version)
CT-89-08 Harry Buhrman, Steven Homer, Leen Torenvliet — Honest Reductions, Completeness and Nondeterminstic Complexity Classes
CT-89-09 Harry Buhrman, Edith Spaan, Leen Torenvliet — On Adaptive Resource Bounded Computations
CT-89-10 Sieger van Denneheuvel — The Rule Language RL/1
CT-89-11 Zhisheng Huang, Sieger van Denneheuvel, Peter van Emde Boas — Towards Functional Classification of Recursive Query Processing

Other Prepublications:

X-89-01 Marianne Kalsbeek — An Orey Sentence for Predicative Arithmetic
X-89-02 G. Wagemakers — New Foundations: a Survey of Quine's Set Theory
X-89-03 A.S. Troelstra — Index of the Heyting Nachlass
X-89-04 Jeroen Groenendijk, Martin Stokhof — Dynamic Montague Grammar, a first sketch
X-89-05 Maarten de Rijke — The Modal Theory of Inequality
X-89-06 Peter van Emde Boas — Een Relationele Semantiek voor Conceptueel Modelleren: Het RL-project

1990 SEE INSIDE BACK COVER