WEAK EQUIVALENCE FOR CONSTRAINT SETS

Sieger van Denneheuvel
Karen Kwast

ITLI Prepublication Series
for Computation and Complexity Theory CT-91-05

University of Amsterdam
The ITLI Prepublication Series

1986
86-01 Peter van Emde Boas
86-02 Johan van Benthem
86-03 Reinhard Muskens
86-04 Kenneth A. Bowen, Dick de Jongh
86-05 Peter van Emde Boas
1987
87-01 Jeroen Groenendijk, Martin Stokhof
87-02 Renate Bartsch
87-03 Willem Klof, Roel van der Vijeer
87-04 Victor Sánchez Valencia
87-05 Eléonore Oversteegen
87-06 Jeroen Groenendijk, Martin Stokhof
87-07 Jeroen Groenendijk
87-08 Renate Bartsch
87-09 Herman Hendriks
1988
LP-88-01 Michiel van Lambalgen
LP-88-02 Yde Venema
LP-88-03 Reinhard Muskens
LP-88-05 Johan van Benthem
LP-88-04 Johan van Benthem
LP-88-07 Renate Bartsch
LP-88-08 Jeroen Groenendijk, Martin Stokhof
LP-88-09 Theo M.V. Jansen
LP-88-10 Anneke Kleppe
ML-88-01 Jaap van Oosten
ML-88-02 M.D.G. Sauer
ML-88-03 Dick de Jongh, Frank Veltman
ML-88-04 A.S. Troelstra
ML-88-05 A.S. Troelstra
CT-88-01 Ming Li, Paul M.B. Vitanyi
CT-88-02 Michiel H.M. Smed
CT-88-03 Michiel H.M. Smed, Mark H. Overmars
ML-88-04 Dick de Jongh, Lex Hendriks
CT-88-05 Peter van Emde Boas
CT-88-06 Michiel H.M. Smed
CT-88-07 Johan van Benthem
CT-88-08 Michiel H.M. Smed, Mark H. Overmars
CT-88-09 Theo M.V. Jansen
CT-88-10 Edith Sann, Leon Torveniet, Peter van Emde Boas
CT-88-11 Sieger van Dennechevel, Peter van Emde Boas
X-88-01 Marc Jumela
LP-89-01 Johan van Benthem
LP-89-02 Jeroen Groenendijk, Martin Stokhof
LP-89-03 Yde Venema
LP-89-04 Johan van Benthem
LP-89-05 Johan van Benthem
LP-89-06 Andrei Prikatelj
LP-89-07 Hendrik Wansing
LP-89-08 Victor Sánchez Valencia
LP-89-09 Zhiheng Huang
ML-89-01 Dick de Jongh, Albert Visser
ML-89-02 Roel van der Vijeer
ML-89-03 Dick de Jongh, Franco Montagna
ML-89-04 Dick de Jongh,Marc Jumela, Franco Montagna
ML-89-05 Rineke Verbruggje
ML-89-06 Michiel van Lambalgen
ML-89-07 Dirk Roorda
ML-89-08 Dirk Roorda
ML-89-09 Alessandra Carbone
CT-89-01 Michiel H.M. Smed
CT-89-02 Peter van Emde Boas
CT-89-03 Ming Li, Herman Neufüllise, Leon Torveniet, Peter van Emde Boas
CT-89-04 Harry Buhrman, Leon Torveniet
CT-89-05 Pierre H. Hartel, Michiel H.M. Smed
CT-89-06 H.W. Lenstra, Jr.
CT-89-07 Ming Li, Paul M.B. Vitanyi
CT-89-08 Harry Buhrman, Steven Homer
CT-89-09 Harry Buhrman, Edith Sann, Leon Torveniet
CT-89-10 Sieger van Dennechevel
CT-89-11 Zhisheng Huang, Sieger van Dennechevel
X-89-01 Marianne Kalbeek
X-89-02 G. Wagenaars
X-89-03 A.S. Troelstra
X-89-04 Jeroen Groenendijk, Martin Stokhof
X-89-05 Maarten de Rijke
X-89-06 Peter van Emde Boas

Other Prepublications:

1990 SEE INSIDE BACK COVER
WEAK EQUIVALENCE FOR CONSTRAINT SETS

Sieger van Denneheuvel
Karen Kwast
Department of Mathematics and Computer Science
University of Amsterdam
Weak equivalence for constraint sets

Sieger van Denneheuel & Karen L. Kwast
Department of Mathematics and Computer Science
University of Amsterdam
Plantage Muidergracht 24, 1018 TV, Amsterdam
Preprint of IJCAI'91 Proceedings of the 12th International Joint Conference on Artificial Intelligence.

Abstract

We describe a generalization of equivalence between constraint sets, called weak equivalence. This new 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. In addition a complete set of axioms for weak implication is presented. With examples derived from the declarative rule language RL we show the applicability of weak equivalence to constraint solving.

1 Introduction

Recently, constraint solving systems have emerged that allow declarative constraint processing (see for example Mathematica [Wolfram, 1988], Bertrand [Leler, 1988] and CLP(\mathbb{R}) [Jaffar and Michaylov, 1987]) whereas more traditional systems required user intervention to direct the solver towards a desired solution. From a general point of view, a typical input specification 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 into two mutually disjoint sets, namely known variables and wanted variables.

Given the above input specification 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 to solve this problem will probably 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.

2 The rule language family RL

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 rule language family RL (see [Van Emde Boas, 1986a, Van Emde Boas, 1986b]). In the RL language, 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 knowledge base system. Query processing should be executed with the help of an existing relational database system; knowledge and queries expressed in RL are to be pre-processed by a constraint solver, and to be compiled into a database query language so that large amounts of data can be processed effectively.

In the current prototype of RL/1 we focus on the integration of a constraint solver with a relational database system; RL/1 has been presented elsewhere [Denneheuel, 1990]. The RL/1 prototype system has been implemented in Prolog which includes a language parser, a database system and a constraint solver. The examples presented in this paper were processed successfully by this prototype.

Terms in RL/1 are constructed inductively in the usual way from constants, variables and functions. Both numeric and string domains are supported. For use in the database application we include binary functions such as \(+, -, *, /\). Constraints include expressions of the form \(t_1 \ op \ t_2\) with \(op \in \{=, >, <, \neq, \leq, \geq\}\) and two special constraints \texttt{false} and \texttt{true}. Null values are not available in the language itself but may be present in the tables of the underlying database (cf. [Kwast, 1991]).

In RL/1 a distinction is made between extensional
objects and intensional objects. Extensional objects correspond to base tables in the underlying relational database. Intensional objects on the other hand are objects whose relations can be materialized by evaluation of their definition. An intensional object is defined by one or more rules, alternatively called clauses. A rule typically consists of a rule head and a rule expression separated by the keyword WHEN. The rule expression contains constraints or invocations of intensional and extensional objects, separated by the conjunctive AND operator. Defining an intensional object with several rules expresses disjunction between the rules. The question whether a rule expression as a whole denotes a finite or infinite relation is in general undecidable; partial information about this question is obtained from the constraint solver.

2.1 Query commands

Query commands in RL/1 result in an answer relation which consists of attributes and a (possibly empty) set of tuples. Optionally the result of a query command can be stored in an extensional table object. There are the following types of query commands:

\[
\text{INFER (attribute-list) WHEN rule-exp [ TO table ]}
\]

\[
\text{SHOW rule-exp [ TO table ]}
\]

The INFER command yields an answer relation with attributes equal to the attribute list between the INFER and WHEN keywords. The SHOW command yields an answer relation whose attributes are the variables occurring in the rule expression.

Processing the above queries requires one or more invocations of the constraint solver. The solver output is used to compile the query into a database request. Symbolic queries on the other hand work by presenting this intermediate solver output to the user directly:

\[
\text{SYMINFER (attribute-list) WHEN rule-exp [ TO file ]}
\]

\[
\text{SYM SHOW rule-exp [ TO file ]}
\]

The ‘TO file’ option allows the computed symbolic answer to be stored in a text file. Symbolic query commands produce one or more of the following answers:

- condition= Condition set
- solution= Solution set

The solution set contains elements of the form \( x = t \) with \( x \) a wanted variable and \( t \) a term. The terms \( f \) in the solution set contain only known variables. The condition part consists of constraints over the known variables; it states what additional restrictions the obtained solution is valid. Intermediate variables are eliminated from both the condition and the solution set. Known variables are determined by the rule expression of the query and wanted variables by the attribute list of the query.

3 Symbolic computation in RL/1

Intensional objects are declarative representations of knowledge; therefore it is not known beforehand how the attributes of an object are invoked. For instance consider the module in Figure 1 for a parallel circuit of two resistors. In the representation of the circuit both the physical laws of basic electricity theory and the practical condition that the resistors should not be overloaded are expressed in terms of constraints. We know from elementary physics that the values given in the table experimentdata fully determine all quantities. How should the RL/1 knowledge base determine that this is indeed the case? The constraint solver is capable to figure out the determinedness.

Suppose voltage is applied to the example circuit then the current through the circuit and the total resistance can be calculated:

\[
\begin{align*}
\text{show circ(9, it, rt, 1200, 400)} \\
\text{rt & it} \\
300 & 0.63
\end{align*}
\]

With other variables given, an answer is also computed:

\[
\begin{align*}
\text{show circ(vt, 0.03, rt, 1200, 400)} \\
\text{vt & rt} \\
9 & 300
\end{align*}
\]

In the constraints for the circuit the law for parallel resistors is not given directly, but the total resistance \( r_t \) can nevertheless be computed. Now suppose we apply various voltages to the two resistors as specified in the experimentdata table in Figure 1. In the experiment the voltage is slowly increased and the resistors stay the same. Surely at some high voltage the circuit will break down. Let’s run the experiment (the ‘(*)’ notation expands to all attributes of the involved object):

\[
\begin{align*}
\text{show experimentdata(*) and circ(*)} \\
\text{vt & r1 & r2 & rt & it} \\
9 & 1200 & 400 & 300 & 0.03 \\
10 & 1200 & 400 & 300 & 0.033
\end{align*}
\]
In the answer relation the rows are corresponding to the rows from the experiment data table but obviously some rows are missing. In these missing rows the constraints were not satisfiable (i.e. the non-smoking condition \(\text{maxcut}(l^2, v)\) was violated) so they do not appear in the answer to the query.

The capability to produce symbolic output is an important feature of the RL/1 knowledge base. It allows the user to analyze relationships between variables and to infer relationships that hold in special circumstances. A general relationship between the resistance of the individual components and the total resistance can be found with the following symbolic query:

\[
\text{syminfer} (rt) \text{ when } \\
\text{known} (r1, r2) \text{ and } \text{cstr}(*) \\
\text{solution} = rt = r1 + r2/(r1+r2)
\]

The constraint solver has inferred the law for parallel resistors.

The classification of variables in the three disjoint partitions of wanted, known and intermediate variables is crucial for derivation of relationships like the previous example. By making \(r_1\) and \(r_2\) known and \(r_1\) wanted the constraint solver of the RL/1 system was directed towards a solution that expresses \(r_1\) in terms of \(r_1\) and \(r_2\). The constraint solver actively applies this information in its inference process to yield the desired result.

On the other hand suppose that variables of a constraint set were classified in only two partitions of wanted and intermediate variables as is the case in many existing constraint solving systems. The constraint solver would correctly establish that the equation system is underdetermined (there are too many variables and not enough equations to solve them).

In the next sections we formalize the notion of equivalence that is used by the solver.

### 4 Weak equivalence and implication

First we introduce some definitions and notations. Variable names are chosen from non capitals (x, y, z, ...) and sets of variables from capitals (X, Y, Z, ...). V is the set of all variables. Constants are denoted by c and functions by f. Terms are denoted by t. Sets of terms are represented as A, B, C or D. To denote the cardinality of a set \(I\) we use the notation \(|I|\).

Solutions are constraints of the form \(x = t\) with \(x \notin \text{var}(t)\). The variable \(x\) is called the head variable and the variables in the term \(t\) the tail 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. \(\{x_1, \ldots, x_n\} \cap \text{var}(\{t_1, \ldots, t_n\}) = \emptyset\)

**Definition 2** A tuple is a solution set of the form \(\{x_1 = c_1, \ldots, x_n = c_n\}\) with \(c_1, \ldots, c_n\) constants.

Solution sets are denoted by \(\Phi\), \(\Psi\) and \(\Theta\) and tuples by \(\phi\), \(\psi\) and \(\theta\). For a solution set \(\Phi = \{x_1 = t_1, \ldots, x_n = t_n\}\) the head variables \(x_1, \ldots, x_n\) are denoted as \(h\Phi(\Phi)\).

The operator restrict retains those solutions in \(\Phi\) that have a head in \(X\):

**Definition 3** \(\Phi[X] = \{x = t \mid x \in X, x = t \in \Phi\}\)

**Definition 4** \(\Phi = x_1 \Psi \text{ iff } \Phi[X] = \Psi[X]\)

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 (collections of) items:

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

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

**Definition 6** (Strong equivalence)
1. \(\phi \models A\) \text{ iff } \forall \phi \in A (\phi = \phi)
2. \(A \models B\) \text{ iff } \forall \phi (\phi \models A \Rightarrow \phi \models B)
3. \(A \equiv E\) \text{ iff } A \equiv B \& B \equiv A

In the case of weak equivalence these definitions are subscripted with the restriction variables \(X\). The notion of strong satisfaction is changed to weak satisfaction by existentially quantifying the variables not in \(X\) (i.e. the intermediate variables). As a consequence the values of these variables in the tuple \(\phi\) are no longer involved in the satisfaction of \(A\).

**Definition 7** (Weak equivalence)
1. \(\phi \models_X A\) \text{ iff } \exists \psi (\psi \models A \& \phi = \psi_X)
2. \(A \models_X B\) \text{ iff } \forall \phi (\phi \models A \Rightarrow \phi \models_B)
3. \(A \equiv_X B\) \text{ iff } A \equiv_X B \& B \equiv_X A

**Example 1** (Equivalence)
\[
\begin{align*}
\{x = 1\} &\not\models_{x,y} \{y = 1\} \quad \{x < x + y, y > 0\} \equiv_x \text{true} \\
\{x = 1\} &\not\models_{x,y} \{y = 1\} \quad \{x < x + y, y > 0\} \equiv_y \{y > 0\} \\
\{x = y\} &\equiv_x \{x = z\} \quad \{x < x + y, y > 0\} \equiv_y \{y > 0\}
\end{align*}
\]

In the above definitions we presented strong implication between constraint sets as an implication using strong satisfaction in both the premise and the consequent. Weak implication was constructed from this definition by weakening the consequent of the implication to weak satisfaction. It may be conceivable that other interesting definitions arise if instead only the premise of the implication is weakened (single implication) or if both the premise and the consequent are weakened (double implication):

**Definition 8** (Single and double implication)
1. \(A \models_X B\) \text{ iff } \forall \phi (\phi \models_X A \Rightarrow \phi \models_B)
2. \(A \models_X B\) \text{ iff } \forall \phi (\phi \models_X A \Rightarrow \phi \models_X B)

In the next proposition we compare these four possible definitions of implication.

**Proposition 9** (Relating implication definitions)
1. \(A \models_X B \Rightarrow A \models B\) \quad (A \models B \neq A \models_X B)
2. \(A \models_B \Rightarrow A \models_X B\) \quad (A \models_X B \neq A \models B)
3. \(A \models_B \Rightarrow A \models_X B\)

The relationships between the four definitions are summarized in Figure 2 (arrows denote implication). Single implication is too strong and only applicable when \(\text{var}(B) \subseteq X\), in which case it is equivalent to strong implication. Double implication is equivalent to weak implication and therefore superfluous.
4.1 Inconsistency, tautology and redundancy

Inconsistency is normally defined as strong equivalence with the special constraint false. As a consequence tautology and redundancy can be defined in terms of strong and weak equivalence. The strong definitions are obtained from the weak definitions by dropping the subscript X.

Definition 10 (Strong and weak inconsistency)
1. Inc_X(A) \iff A \equiv_X false
2. Taut_X(A) \iff A \equiv_X true
3. Red_X(A) \iff \exists B \subset A (||B|| < ||A||) & A \equiv_X B

It turns out that weak and strong inconsistency are actually equivalent:

Proposition 11
1. A \equiv_X false \iff A \equiv false
2. A \equiv_X false \iff A \equiv false
3. Inc(A) \iff Inc_X(A)

However, contrary to inconsistency, weak tautology is truly weaker than strong tautology:

Proposition 12
1. Taut(A) \Rightarrow Taut_X(A)
2. Taut_X(A) \neq Taut(A)

It is easy construct a counter example such that Taut_X(A) but not Taut(A). Let A = \{x < x + y, y > 0\} \equiv \{x\} true so Taut_X(A). On the other hand \neg Taut(A) because \{x < x + y, y > 0\} \equiv \{y > 0\} \neq true.

As a consequence weak redundancy is not equivalent to strong redundancy:

Proposition 13
1. Red(A) \Rightarrow Red_X(A)
2. Red_X(A) \neq Red(A)

Again we can construct a counter example such that Red_X(A) but not Red(A). Let A = \{x = y + 2, y = z + 2\} then it holds that A \equiv_{(x,y)} \{x = y + 2\} so Red_X(A). On the other hand \neg Red(A) because both \{x = y + 2, y = z + 2\} \neq \{x = y + 2\} and \{x = y + 2, y = z + 2\} \neq \{y = z + 2\}.

5 Weak axioms

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

Proposition 14 (The equivalence relation \equiv_X)
1. A \equiv_X A
2. A \equiv_X B & B \equiv_X C \Rightarrow A \equiv_X C
3. A \equiv_X A
4. A \equiv_X B \Rightarrow B \equiv_X A
5. A \equiv_X B & B \equiv_X C \Rightarrow A \equiv_X C

Now we explore some axioms that are specifically concerned with the restriction variables of weak implication. As was to be expected, restriction variables can be harmlessly deleted:

Proposition 15 (Removal of restriction variables)
1. A \equiv_{X \cup Y} B \Rightarrow A \equiv_B B
2. A \equiv_{X \cup Y} B \Rightarrow A \equiv_B B

A variable can be added to the set of restriction variables X in A \equiv_B B if it does not occur in B:

Proposition 16 (Addition of restriction variables)
1. A \equiv_X B \Rightarrow A \equiv_{X \cup Y} B if \text{var}(B) \cap Y \subset X
2. A \equiv_X B \Rightarrow A \equiv_{X \cup Y} B if \text{var}(A \cup B) \cap Y \subset X

For strong implication we have, among others, the following axioms:

Proposition 17 (Axioms for strong implication)
1. A \cup B \equiv_A A
2. A \equiv B & A \equiv C \Rightarrow A \equiv B \cup C
3. A \equiv B \Rightarrow A \cup C \equiv B \cup C
4. A \equiv B \cup C \Rightarrow A \equiv B
5. A \equiv B & B \equiv C \equiv D \Rightarrow A \equiv C \equiv D
6. A \equiv B \cup C & B \equiv D \Rightarrow A \equiv C \equiv D
7. A \equiv B & C \equiv D \Rightarrow A \equiv C \equiv B \equiv D

The axioms (1) and (2) are known in Functional Dependency Theory [Vardi, 1988] as decomposition and union respectively. From these axioms and transitivity (axiom (2) of Proposition 14) all other strong axioms (3)-(7) can be derived. For weak implication however the situation is different, since only (1) and (4) hold unconditionally:

Proposition 18 (Axioms for weak implication)
1. A \cup B \equiv_X A
2. A \equiv_X B & A \equiv_X C \Rightarrow A \equiv_X B \cup C
3. A \equiv_X B \Rightarrow A \cup C \equiv_X B \cup C
4. A \equiv_X B \cup C \Rightarrow A \equiv_X B
5. A \equiv_X B & B \equiv_X C \equiv_X D \Rightarrow A \equiv_X B \cup C \equiv_X D
6. A \equiv_X B \equiv_X C \Rightarrow A \equiv_X C \equiv X
7. A \equiv_X B & C \equiv_X D \Rightarrow A \equiv_X C \cup B \equiv_X D

The conditional axioms of Proposition 18 are more general than their counterparts in Proposition 17 since from the conditional axioms the unconditional axioms can be derived. For instance axiom (2) in Proposition 17 can be derived from axiom (2) in Proposition 18 by assigning to X the set of all variables V in which case the condition becomes trivially true.
6 Completeness

A natural question that arises in the context of an axiomatic approach is whether a sound and complete axiomatization is feasible. As a first step we proved the soundness of all axioms in Figure 3.

Proposition 19 For general constraints, the axioms in Figure 3 are sound.

Both for weak and strong implication, a completeness result can be derived. We proved completeness with respect to primitive constraints, that is constraints of the format \( x = y \) or \( x = c \):

Proposition 20 For primitive constraints, the axioms in Figure 3 are sound and complete.

For completeness the axioms concerned with substitution have to be added to the ones we gave before. The most interesting axiom is Abstraction that allows you to add solutions for intermediate variables.

From the complete set for weak implication a complete set of strong axioms can be easily inferred. Abstraction is never applicable for strong implication so it is replaced by the derived axiom Instantiation:

\[ A \models B \Rightarrow \Phi(A) \models \Phi(B) \]

7 Constraint elimination

Intuitively more constraints can be eliminated if weak equivalence is used instead of strong equivalence. In this section we provide a proposition that specifically exploits weak equivalence for constraint elimination. A subset \( B \) of a set of constraints \( A \cup B \) is redundant with respect to the restriction variables if there exists a solution set \( \Phi \) for intermediate variables such that \( \Phi(B) \), the effect of substituting \( \Phi \) on \( B \), is equivalent with true:

Proposition 21

\[ A \cup B \equiv_X A \text{ if there exists some } \Phi \text{ such that:} \]

1. \( \Phi(B) \models \text{true} \)
2. \( \var(B) \cap X = \emptyset \)
3. \( \var(B) \cap \var(A) = \emptyset \)

Proof:

i) \( A \cup B \models_X A \): Weakening.

ii) \( A \models_X A \cup B \):

\[ A \models_X A \]

\[ \Rightarrow A \models_X \Phi(A) \] \hspace{1cm} (3)

8 An elaborated example

As an illustrative example of application of weak equivalence consider the circuit of Figure 4 derived from the domain of electronics. The corresponding RL/1 representation is listed in Figure 5. In ancient history the bridge of Wheatstone was used to measure ohm resistance. In the setup, \( r_1 \) (or \( r_2 \)) is a variable resistor and \( r_4 \) (or \( r_3 \)) the unknown resistor. The component in the center is an ammeter with resistance \( r_5 \). For measurement the variable resistor is adjusted so that no current flows through \( r_5 \). In this balanced state the following relationship holds:

\[ r_1/r_2 = r_3/r_4 \]

To see if theory matches reality we apply 10 volt at the clamps of the circuit. In case we choose the values of the resistors all equal to 200 ohm then the circuit is in balanced state and the RL/1 knowledge base can compute the values for \( v_b \) and \( v_c \):

\[
\text{show circ}(10, v_b, v_c, 0, 15, 200, 200, 200, 200, 200) \\
v_b \quad v_c \quad 15 \\
5.0000 \quad 5.00000 \quad 0
\]

To obtain this answer, the solver infers the following weak equivalence:
These equations are nonlinear by the occurrences of the terms $i_1 \cdot r_1$, $i_2 \cdot r_2$, $i_3 \cdot r_3$, $i_4 \cdot r_4$ and $i_5 \cdot r_5$ (note that the known variables $r_1$, $r_2$ and $r_3$ are in fact not given as a value and therefore contribute to the nonlinearity of the constraint system).

9 Conclusion

In this paper we used weak equivalence to express, explicitly, the functional difference between wanted, known and intermediate variables in a set of constraints. By a complete set of axioms we established the main properties of weak equivalence. Using these axioms we proved an application independent tool to simplify constraint sets using this new form of equivalence.

With examples run on a prototype implementation we showed practical applicability of weak equivalence for constraint solving purposes. By distinguishing wanted, known and intermediate variables in a constraint set the solver can be guided towards a solution that discards intermediate variables and expresses wanted variables in terms of known variables. Derivation of symbolic relationships is facilitated by the theoretical concepts introduced in this paper in two ways. Firstly, weak equivalence allows you to express formally the correspondence between the derived relationship and the original constraint set. Secondly the reasoning necessary for inference of symbolic relationships can be represented in terms of weak equivalence.

References


The ITLI Prepublication Series

1990

Logic, Semantics and Philosophy of Language
LP-90-01 Jaap van der Does
LP-90-02 Jeroen Groenendijk, Martin Stokhof
LP-90-03 Reinhard Lotz
LP-90-04 Arme Ranta
LP-90-05 Patrick Blackburn
LP-90-06 Gennaro Chierchia
LP-90-07 Gennaro Chierchia
LP-90-08 Herman Hendriks
LP-90-09 Paul Dekker
LP-90-10 Theo M.V. Janssen
LP-90-11 Johan van Benthem
LP-90-12 Serge Lapierre
LP-90-13 Zhisheng Huang
LP-90-14 Jeroen Groenendijk, Martin Stokhof
LP-90-15 Maarten de Rijke
LP-90-16 Zhisheng Huang, Karen Kwast
LP-90-17 Paul Dekker

Mathematical Logic and Foundations
ML-90-01 Harold Schellinx
ML-90-02 Jaap van Oosten
ML-90-03 Yde Venema
ML-90-04 Maarten de Rijke
ML-90-05 Domenico Zambella
ML-90-06 Jaap van Oosten
ML-90-07 Maarten de Rijke
ML-90-08 Harold Schellinx
ML-90-09 Dick de Jongh, Duccio Pianigiani
ML-90-10 van Lambalgen
ML-90-11 Paul C. Gilmore

Computation and Complexity Theory
CT-90-01 John Tromp, Peter van Emde Boas
CT-90-02 Sieger van Denneheuvel, Gerard R. Renardel de La Valette
CT-90-03 Richard Gavaldà, Leen Torenvliet
CT-90-04 Osamu Watanabe, José L. Balcazar
CT-90-05 Sieger van Denneheuvel, Karen Kwast
CT-90-06 Michel Smid, Peter van Emde Boas
CT-90-07 Kevin Verboquet
CT-90-08 Fred de Geus, Ernest Rotterdam
CT-90-09 Sieger van Denneheuvel, Peter van Emde Boas
CT-90-10 Roel de Vrijer

Other Prepublications
LP-90-01 Yde Venema
X-90-01 A.S. Troelstra
X-90-02 Maarten de Rijke
X-90-03 L.D. Beklemishev
X-90-04 Annual Report 1989
X-90-05 Valentin Shehtman
X-90-06 Valentin Goranko, Solomon Passy
X-90-07 V.V. Shavrukov
X-90-08 L.D. Beklemishev
X-90-09 V.Yu. Shavrukov
X-90-10 Sieger van Denneheuvel, Peter van Emde Boas
X-90-11 Alessandro Carboni, I. Lázár
X-90-12 Maarten de Rijke
X-90-13 K.N. Ignatiev
X-90-14 L.A. Chagrova
X-90-15 A.S. Troelstra

Mathematical Logic and Foundations
ML-91-01 Alessandro Berarducci, Rineke Verbrugge
ML-91-02 Domenico Zambella
ML-91-03 Raymond Hoofman, Harold Schellinx
ML-91-04 Raymond Hoofman, Harold Schellinx
ML-91-05 A.S. Troelstra
CT-91-01 Ming Li, Paul M.B. Vitanyi
CT-91-02 Ming Li, John Tromp, Paul M.B. Vitanyi
CT-91-03 Ming Li, Paul M.B. Vitanyi
X-91-01 Alexander Chagrov, Michael Zakharyaschev
X-91-02 Alexander Chagrov
X-91-03 V. Yu. Shavrukov
X-91-04 K.N. Ignatiev
X-91-05 Johan van Benthem
X-91-06 Annual Report 1990
X-91-07 A.S. Troelstra

1991

Mathematical Logic and Foundations
ML-91-01 Yde Venema
ML-91-02 Domenico Zambella
ML-91-03 Raymond Hoofman, Harold Schellinx
ML-91-04 Raymond Hoofman, Harold Schellinx
ML-91-05 A.S. Troelstra
CT-91-01 Ming Li, Paul M.B. Vitanyi
CT-91-02 Ming Li, John Tromp, Paul M.B. Vitanyi
CT-91-03 Ming Li, Paul M.B. Vitanyi
X-91-01 Alexander Chagrov, Michael Zakharyaschev
X-91-02 Alexander Chagrov
X-91-03 V. Yu. Shavrukov
X-91-04 K.N. Ignatiev
X-91-05 Johan van Benthem
X-91-06 Annual Report 1990
X-91-07 A.S. Troelstra

Other Prepublications
X-91-01 Alexander Chagrov, Michael Zakharyaschev
X-91-02 Alexander Chagrov
X-91-03 V. Yu. Shavrukov
X-91-04 K.N. Ignatiev
X-91-05 Johan van Benthem
X-91-06 Annual Report 1990
X-91-07 A.S. Troelstra

The Disjunction Property of Intermediate Propositional Logics
On the Undecidability of the Disjunction Property of Intermediate Logics
Subalgebras of Diagonizable Algebras of Theories containing Arithmetic
Partial Conservativity and Modal Logics
Temporal Logic
Lectures on Linear Logic, Errata and Supplement