A Horn Clause that Implies an Undecidable Set of Horn Clauses *

Jerzy Marcinkowski
Institute of Computer Science
University of Wrocław,
jma@ii.uni.wroc.pl

Abstract

In this paper we prove that there exists a Horn clause $\mathcal{H}$ such that the problem: given a Horn clause $\mathcal{G}$. Is $\mathcal{G}$ a consequence of $\mathcal{H}$? *is not recursive. Equivalently, there exists a one-clause PROLOG program such that there is no PROLOG implementation answering TRUE if the program implies a given goal and FALSE otherwise. We give a short survey of earlier results concerning clause implication and prove a classical Linial-Post theorem as a consequence of one of them.*

1 Introduction

1.1 Introduction

Our main interest is the analysis of the decidability of the implication problem $\mathcal{H}_1 \Longrightarrow \mathcal{H}_2$ where $\mathcal{H}_1$ and $\mathcal{H}_2$ are Horn clauses in the language of the first order logic without equality.

We adopt the following taxonomy: A literal is an atomic formula (an atom) or its negation. An atom is of the form $Q(t_1, t_2, \ldots t_k)$ where Q is a k-ary predicate symbol and t's are first order terms constructed from function symbols, variables and constants. A clause is a *universal closure* of a disjunction of literals. A Horn clause (or a program clause) is a clause with at most one positive (i.e. not negated) literal. A Horn clause with k negative and one positive literal will be called a k-clause (here we do not obey the standard that a k- clause is a disjunction of k literals). A Horn clause can be in the usual way written in an implication form.

A term is ground if it contains no variables. An atom is ground if all its arguments are ground terms. A substitution is a finite function σ such that the arguments of σ are variables and values are terms. σ can be extended to act on terms, literals and clauses. Terms t_1 and t_2 are unifiable if there exists a substitution σ such that $\sigma(t_1) = \sigma(t_2)$.

If $\mathcal{H} = (A_1 \wedge A_2 \wedge \ldots \wedge A_k \Longrightarrow A)$ is a k-Horn clause then a $\mathcal{H}$ -derivation is a full k-tree $\mathcal{D}$ (i.e. such that every its node is a leaf or has exactly k sons) with each its node labelled by a ground atom so that, if w is an inner node of $\mathcal{D}$, Δ is the label of w and for w_i, the i'th son of w, Δ_i is the label of the son then there exists a substitution σ such that $\sigma(A) = \Delta$, and $\sigma(A_i) = \Delta_i$ for $i = 1, 2, \ldots k$.

The following lemmas allow us to use two equivalent languages to talk about the problems of our interest: one of them will be the language of clause implication and the other the one of derivations:

* This research has been supported by a KBN grant.

Lemma 1.1 *Let $\mathcal{H}_1$ and $\mathcal{H}_2$ be Horn clauses and let $\mathcal{H}_2 = (A_1 \wedge A_2 \wedge \ldots \wedge A_k \Longrightarrow A)$. Then $\neg(\mathcal{H}_1 \Longrightarrow \mathcal{H}_2)$ holds iff the set consisting of $\mathcal{H}_1$, $\{\sigma(A_i) : i = 1, 2, \ldots k\}$ and $\neg\sigma(A)$ is satisfiable, where σ substitutes a constant c_x, occuring neither in $\mathcal{H}_1$ nor in $\mathcal{H}_2$, for every variable x in $\mathcal{H}_2$.*

Proof: The *if* implication is obvious. To prove the opposite implication consider a model $\mathcal{M}$ in which $\mathcal{H}_1$ is true and $\mathcal{H}_2$ is not. So, there is a valuation v of the variables in $\mathcal{H}_2$ such that every $\mathsf{v}(A_i)$ is true in $\mathcal{M}$ but $\mathsf{v}(A)$ is not. Put $\mathsf{v}(x)$ as the interpretation of c_x in $\mathcal{M}$ for every x in $\mathcal{H}_2$. ■

Lemma 1.2 *Let $\mathcal{H}_1$ and $\mathcal{H}_2$ be Horn clauses and let $\mathcal{H}_2 = (A_1 \wedge A_2 \wedge \ldots \wedge A_k \Longrightarrow A)$ and let $\sigma(x) = c_x$, for every variable x in $\mathcal{H}_2$. Then $\mathcal{H}_1 \Longrightarrow \mathcal{H}_2$ iff there exists a $\mathcal{H}_1$ derivation with the leaves labelled by elements of $\{\sigma(A_i) : i = 1, 2, \ldots k\}$ and with the root labelled with $\sigma(A)$.*

Proof: Clearly our notion of $\mathcal{H}$ -derivation coincides with the notion of SLD-resolution. So the lemma follows from the Lemma 1.1., and from the completeness of SLD-resolution (see[1]). ■

1.2 Clause Implication: Earlier Results

Questions concerning decidability of clause implication and similar problems are motivated by artificial intelligence and automated deduction and they have been an area of some interest recently. Redundancy elimination is known to be an important issue in optimization of automated theorem provers (where general clauses are used) or logic programs and clausual knowledge bases (where all clauses are Horn clauses). Less redundant clause sets require less storage space, but first of all, since the redundant clauses force the program to do redundant work, allow better performance of proof procedures.

In practice, in automated theorem provers, the subsumption of clauses, not the clause implication, is used as a criterion of deletion. We say that the clause $\mathcal{H}$ subsumes the clause $\mathcal{G}$ if there exists a substitution σ such that $\sigma(\mathcal{H}) \subseteq \mathcal{G}$ (as a set of literals). Of course, if $\mathcal{H}$ subsumes $\mathcal{G}$ then also $\mathcal{H}$ implies $\mathcal{G}$. The problem whether the clause $\mathcal{H}$ subsumes the clause $\mathcal{G}$ is decidable and, what is much more, is NP-complete. Subsumption algorithms were proposed in [5] and [21]. Their complexity analysis and some new algorithms can be found in [8] and [9].

It is easy to observe that it is not always so, that if a clause $\mathcal{H}$ implies clause $\mathcal{G}$ then $\mathcal{H}$ subsumes $\mathcal{G}$. Hence there are cases of redundancy that can be recognized by the clause implication test but not by the clause subsumption test. Unfortunately the problem if (general) clause $\mathcal{H}$ implies clause $\mathcal{G}$ has been proved to be undecidable by M. Schmidt-Schauss in 1988[22]. His theorem can be formulated also as: the problem, *given three Horn clauses, do the first two imply the third ?* is undecidable. The decidability of a clause implication problem for Horn clauses (i.e. whether a Horn clause $\mathcal{H}$ implies a Horn clause $\mathcal{G}$) remained open until 1992, when it was solved negatively by J. Marcinkowski and L. Pacholski ([19]).

Meanwhile, some strong sufficient conditions for decidability of Horn clause implication where given ([10],[22],[15],[16]). We follow[10] and present a survey of the results here:

(i) If $\mathcal{H}$ ranges over unary Horn clauses then the problem if $\mathcal{H}$ implies a Horn clause $\mathcal{G}$ is decidable. Equivalently, the problem, if there exists an $\mathcal{H}$ -derivation

with its premise (notice that such a derivation has just one premise) and conclusion labelled with given ground atoms, is decidable. We are not able to name the author of the theorem here, however following [22] we shall present a proof of it.

Proof: Let $\mathcal{H} = (Q(t_1) \Longrightarrow Q(t_2))$. Let $Q(s_1)$ be the (hypothetical) label of the premise of the derivation and $Q(s_2)$ be the one of the conclusion (s_1 and s_2 are ground terms). Following[22] we define $\mathcal{H}_1 = \mathcal{H}$. Let $\mathcal{H}_k = (Q(r_1^k) \Longrightarrow Q(r_2^k))$ (we assume that the variables in r's are distinct from those in t's). Let σ be the most general substitution such that $\sigma(t_1) = \sigma(r_2^k)$. Then define $\mathcal{H}_{k+1} = Q(\sigma(r_1^k)) \Longrightarrow Q(\sigma(t_2))$. Notice, that $\mathcal{H}_{k+1}$ would be the same (up to renaming of variables) if we have defined σ as the most general substitution such that $\sigma(t_2) = \sigma(r_1^k)$. and $\mathcal{H}_{k+1}$ as $Q(\sigma(r_2)) \Longrightarrow Q(\sigma(t_1))$. Let d be a natural number greater than the depth of s_1 and s_2 . There are two possibilities: Either for every k the depth of literals (to be more precise, of terms) of $\mathcal{H}_k$ is smaller than d. So, since there are only finitely many literals of depth smaller than d the function $k \mapsto \mathcal{H}_k$ is periodical for sufficiently large k. Or there exists a k such that one of the terms of $\mathcal{H}_k$ is deeper than d. Consider the smallest of such k's. Suppose r_1^k is deeper than d (if not, r_2^k is, and the argument is symmetrical). Then every r_1^l $(l > k)$ is an instance of r_1^k and thus is deeper than d. In both cases the set $\Xi = \{\mathcal{H}_k$:the depth of its literals is smaller than $d\}$ is finite and computable.

Now, if there exists $\mathcal{H}_k \in \Xi$ such that $\mathcal{H}_k = Q(r_1) \Longrightarrow Q(r_2)$ and a substitution σ such that $\sigma(r_1) = s_1$ and $\sigma(r_2) = s_2$ then (and only then) $\mathcal{H}$ implies $Q(s_1) \Longrightarrow Q(s_2)$. The last question is decidable. ■

(ii) In this and next items we will assume that $\mathcal{H} = (A_1 \wedge A_2 \wedge \ldots \wedge A_k \Longrightarrow A)$. Suppose there is no such i and a substitution σ such that $\sigma(A_i) = \sigma(A)$ (We say that $\mathcal{H}$ is not a recursive clause). Then $\mathcal{H}$ implies a clause $\mathcal{G}$ iff $\mathcal{G}$ is a tautology or if $\mathcal{H}$ subsumes $\mathcal{G}$. So the implication problem is decidable.

(iii) If $\mathcal{H}$ is a linearly recursive clause (i.e. if there is only one i for which there exists a substitution σ that $\sigma(A_i) = \sigma(A)$) then the problem if $\mathcal{H}$ implies a clause $\mathcal{G}$ is decidable. It is similar to the problem proved to be decidable in (i) ([22],[15]).

(iv) The Horn clause implication problem is also decidable if each variable occuring in the conclusion of $\mathcal{H}$ occurs on a deeper position in some premise of $\mathcal{H}$ and the conclusion does not exceed any premise in terms depth ([15]).

(v) The Horn clause implication problem is decidable if each variable of $\mathcal{H}$ occurs in such a premise A_i that there is no such substitution σ that $\sigma(A_i) = \sigma(A)$ ([15]).

(vi) If $\mathcal{H} = (A_1 \wedge A_2 \wedge \ldots \wedge A_k \Longrightarrow A)$ is such that A contains at most one variable and there is an i such that the premise A_i contains all variables of all premises then the clause implication problem is decidable for an arbitrary clause $\mathcal{G}$ ([10]).

(vii) Let $\mathcal{H} = (A_1 \wedge A_2 \wedge \ldots \wedge A_k \Longrightarrow A)$ be a Horn clause such that there exists i such that A_i contains all variables occuring in A and such that A and $\sigma(A_i)$ are non unifiable, where σ substitutes new variables for those in A_i. The problem if $\mathcal{H}$ implies an arbitrary clause $\mathcal{G}$ is decidable ([10]).

1.3 A Related Topic: Cycle Unification Problem

The problem (i) proved to be decidable in the previous section is the strongest of the natural decidable problems in the area of clause implication. It has two natural generalizations: first of them is to admit more than only one premise. We get the full Horn Clause Implication Problem then, which is the main topic of this paper. The second is the so called Cycle Unification Problem: *given an unary*

Horn clause $\mathcal{H} = (Q(t_1) \Longrightarrow Q(t_2))$ *and two* **arbitrary** *terms* s_1 *and* s_2*, is there any substitution* σ *such that* $Q(\sigma(s_2))$ *is a logical consequence of* $\mathcal{H}$ *and* $Q(\sigma(s_1))$*?* Similarly as in the case of Horn Clause Implication Problem some papers have been written, giving sufficient conditions for decidability of the Cycle Unification Problem. The survey of the results and proofs can be found in[3]. The general problem was proved to be undecidable in 1992 by P. Devienne, P. Lebégue and J.C. Routier ([6],[7]). In 1993 P. Hanschke and J. Würz ([23],[12]) gave a new, very clever and simple proof of the undecidability.

It is nice to see that this result can provide a new and easy proof of a classical Linial-Post theorem ([17]). Consider a language of terms built of variables and one binary functional symbol $\Longrightarrow$. Suppose terms in some finite set $\mathcal{S}$ are already considered as proved. Then, we can use one of two rules to enrich the set of proved terms. First, if t is proved and σ is a substitution then also $\sigma(t)$ is proved. Second, if t, s are proved, $s = (t \Longrightarrow r)$ then also r is proved (first of those two is the substitution rule and the second is modus ponens. When writing about the use of modus ponens we will call s a modus ponens major premise and t a modus ponens minor premise). If, starting with some set $\mathcal{S}$ we can, applying the above rules finitely many times, get a term s then we write $\mathcal{S} \vdash s$.

Theorem 1.3 (Linial, Post) *The problem*

Given a set $\mathcal{S}$ of tautologies of the propositional calculus and a term s that is a tautology. Does $\mathcal{S} \vdash s$?

is undecidable.

This theorem gives an answer to a classical Tarski's question about decidability of derivability in propositional calculus. The proof of the theorem can be found in [20] and in [24]. It is elementary but quite complicated. In a short Section 3 we prove it as a corollary of the undecidability of Cycle Unification Problem.

1.4 Open Problems and Finite Implication

There are, as far as we know, no more natural open problems concerning decidability of clause implication, at least when we consider the classical meaning of the notion of implication.

However there are still many theorems left to be proved in the case of finite implication. It can be argued that the word described by a knowledge base is always a finite set of objects. So, we can claim that the example of a natural question is not the decidability of the problem *does there exist* **any** *structure in which a Horn clause* $\mathcal{H}$ *is satisfied and a Horn clause* $\mathcal{G}$ *is not ?* but rather of the one *does there exist a* **finite** *structure in which a Horn clause* $\mathcal{H}$ *is satisfied and a clause* $\mathcal{G}$ *is not?* If the last is not true, then we say that $\mathcal{H}$ finitely implies $\mathcal{G}$.

Observe that the proof of the Lemma 1.1. is valid for the clause finite implication case, as well as for the classical one. So we can equivalently ask the decidability questions in the language of clause implication and in the language of logic programs. It is however clear that there are no reasons for the Lemma 1.2. to be valid for the finite case. So, we can not, as we do in our Main Results sections, use the language of derivations to consider problems of finite implication.

There are some similarities between decidability problems of clause implication and decidability problems of dependencies implication in relational database theory (see [14])(but in fact, as we know up to now, the similarities are rather formal than technical). The language of first order logic **with** equality with no

function symbols and with one n-ary predicate symbol A is used there. A dependency is a sentence

$$\forall y_1, \ldots, y_k \exists x_1, \ldots, x_l (A(\mathbf{t_1}) \wedge A(\mathbf{t_2}) \wedge \ldots A(\mathbf{t_p})) \Longrightarrow (B(\mathbf{s_1}) \wedge B(\mathbf{s_2}) \wedge \ldots B(\mathbf{s_q}))$$

where each B is either A or $=$, $\mathbf{s_i}, \mathbf{t_j}$ denote vectors of variables of proper length, the variables occuring in the premises of the implication are exactly $y_1, y_2 \ldots, y_k$ and the set of variables occuring in the conclusions contains $x_1, x_2, \ldots, x_l$. The (finite) implication problem for data dependencies is the problem of deciding whether a given set of dependencies logically (finitely) implies another dependency. Many sorts of dependencies have been considered and some results concerning undecidability of finite implication have been proved. Usually the following two results have been used as tools in the proofs of undecidability:

Theorem 1.4 *An equation is a sentence $\forall y_1, \ldots, y_k (s = t)$ where s, t are terms of first order logic with one functional symbol f and without constants. The problem*

Given an equation. Does it have a finite non-trivial (i.e. consisting of more than one element) model?

is undecidable.

Theorem 1.5 *Let $\mathcal{A}$ be an alphabet. $\mathcal{A}^*$ is a free monoid generated by $\mathcal{A}$. Let $E = \{a_i = b_i\}$ be a finite set of equations of words in $\mathcal{A}^*$ and $a = b$ be another such equation. The problem:*

Given E, a, b. Does there exist a finite monoid M and a homomorphism $h : \mathcal{A}^* \longrightarrow M$ such that $h(a_i) = h(b_i)$ for all i and $h(a) \neq h(b)$?

is undecidable.

Theorem 1.4. is due to R.McKenzie ([18]) and has been a main tool in the proof of undecidability of the finite implication in [2]. Theorem 1.5. , due to Y. Gurevich ([11]) was used in [4] and [13].

We report the topics of database dependencies here since we hope, that the two theorems used there can be also good tools for our needs in the case of finite implication. Unfortunately, the equality symbol seems to be inherently present in both of them. In fact we do not know any question similar to our in the language without equality. It is not clear how to encode the equality, or something sufficiently similar to it, within one Horn clause in the language without the equality symbol.

1.5 The Results

In the Section 2, *The Main Result* we give a proof of the following theorem:

Theorem 1.6 *(i) There exist a Horn clause $\mathcal{H}$ and a finite set $\mathcal{S}$ of ground atoms such that the problem:*
Given a ground atom A. Does there exist a finite $\mathcal{H}$ -derivation with its leaves labelled by elements of $\mathcal{S}$ and with its root labelled by A ?
is undecidable.

(ii) There exist a clause $\mathcal{H}$ and atoms $A_1, A_2, \ldots A_k$ such that the problem
Given an atom A . Does $\mathcal{H}$ imply $\mathcal{G} = (A_1 \wedge A_2 \wedge \ldots \wedge A_k \Longrightarrow A)$?
is undecidable

Parts (i) and (ii) of the theorem are equivalent by Lemma 1.1. and Lemma 1.2. The theorem is stronger than the result of [19] where the following was proved:

Theorem 1.7 *(i) The problem*
Given a Horn clause $\mathcal{H}$, a set $\mathcal{S}$ of ground atoms and a ground atom A. Does there exist a finite $\mathcal{H}$ -derivation with its leaves labelled by elements of $\mathcal{S}$ and with the root labelled by A ?
is undecidable.

(ii) The problem
Given Horn clauses $\mathcal{H}$ and $\mathcal{G}$. Does $\mathcal{H}$ imply $\mathcal{G}$?
is undecidable

As in Theorem 1.6. Parts (i) and (ii) of the theorem are equivalent by Lemma 1.1. and Lemma 1.2.. The difference between results of Theorem 1.6. and Theorem 1.7., although perhaps not very important, still can be viewed as quite practical. We can think that logic programming is an activity of three participants: First of them is the company that produces the software for compiling and executing a logic program (e.g. PROLOG). Second is the author of the program itself. So it is a person or a team that writes the clause (clauses, but recall that all of that was already known for programs containing more than one clause) and the set of facts (which we denote by $\mathcal{S}$). Third participant is the user of the program which asks it some queries and hopes to get an answer TRUE if the program implies the query and FALSE otherwise. Our former result have asserted that the job of the second participant is not very easy: he can not buy a proper software in a shop and next use it to all programs he writes but rather he must prepare it himself, depending of the form of the program he has written. The new result tells that in some cases the job of the second participant is not only difficult but just impossible to be done.

The method of the proof in this paper does not differ much from the one of [19], but the understanding of the method is deeper here. In the former paper we have built a clause $\mathcal{H}$ such that the label of a node of a $\mathcal{H}$ -derivation was equal (as a tree) to the part of the derivation rooted in the node and such that the problem *does there exist a finite derivation containing a given node* w was undecidable. The problem was how to force the derivation to contain w . In the former paper we gave a solution: we changed $\mathcal{H}$ in a way depending of w. Here only the conclusion is used to force the derivation to contain the required node.

2 The Main Result

2.1 Preliminaries

In this subsection we give basic definitions concerning Turing machines, Thue processes, derivations and finite trees, and we formulate some lemmas about connections between the defined notions.

Definition 2.1 *Let* c_1 *and* c_2 *be configurations of a deterministic Turing machine* T. *Then* $c_1 \to c_2$ *denotes the fact, that* T *moves from* c_1 *to* c_2 *in one step. Let* $\mathcal{O}(c)$ *denote the equivalency class of* c *with respect to the symmetric, reflexive and transitive closure of* $\to$. T *is* useful *if it satisfies the following conditions:*

(i) the tape of T *is right-infinite.*

(ii) the tape symbols of T *are $, !, 0 and 1.*

(iii) ! and $ are special symbols: machine can neither write nor delete them. ! marks the (left) end of the tape, so T *has no instruction of the form: "while scanning ! in the state q move left".*

(iv) denote by N *the following configuration of* T *:* $!0^{n-2}\$$ *on the tape, head in the initial state over the symbol \$. It is undecidable, if* $\mathcal{O}$(N) *is finite.*

Consider the set $\mathcal{A}$ of triples $\langle$*tape symbol*, *state*, *end of the tape*$\rangle$ where *tape symbol* is one of 0,1,!,\$, *state* is one of the states of T or 0 and *end of the tape* is 0 or 1. We will think of $\mathcal{A}$ as of an alphabet. The triples in $\mathcal{A}$ will be symbols of the alphabet. Symbols $\langle \$, ., . \rangle$ will be called *dollars* and symbols $\langle ., ., 0 \rangle$ will be called *static symbols.* Symbols that are not static will be called *dynamic.* Symbols $\langle ., q, . \rangle$ where $q \neq 0$ we will call *headmarking.* Notice, that each configuration of T can be uniquely identified with a word w over $\mathcal{A}$ such that only one symbol of w is headmarking and only one, last symbol of w, is dynamic. Let $\mathcal{T}$ be a Thue system with alphabet $\mathcal{A}$.

Definition 2.2 *For every word w over* $\mathcal{A}$ *we put thue*$(w) = \{v : v \overset{*\mathcal{T}}{\Longleftrightarrow} w\}$.
For every set $\mathcal{W}$ *of words over the alphabet* $\mathcal{A}$ *we put THUE*$(\mathcal{W}) = \{v : \exists w \in \mathcal{W}$ *\exists u: v is a prefix of u and* $w \overset{*\mathcal{T}}{\Longleftrightarrow} u\}$.

Definition 2.3 *A Thue process* $\mathcal{T}$ *is* useful *if for every w thue(w)=*$\mathcal{O}$*(w) and:*
(i) if $< u, v >$ *is a production of* $\mathcal{T}$ *then each of u and v has no more than 2 symbols and no less than one symbol. u and v are not both single symbols.*
(ii) if a symbol $i \in \mathcal{A}$ *occurs in any production* $< i, w >$ *then it is dynamic.*
(iii) if a symbol i occurs in any production of a form $< ji, k >$ *where* j, k *are symbols, then it is dynamic.*
(iv) if a symbol i occurs in any production of the form $< ij, v >$ *then it is static.*
(v) if $< ij, kl >$ *is a production of* $\mathcal{T}$ *and j is dynamic, so is l.*
(vi) if $< ij, kl >$ *is a production of* $\mathcal{T}$ *and j is dollar, so is l.*
(vii) if $< ij, kl >$ *is a production of* $\mathcal{T}$ *and i is dollar, so is k.*

Lemma 2.4 *There exists a useful Turing machine* T *and useful Thue process* $\mathcal{T}$.

Proof: It is clear that we can construct a universal Turing machine T satisfying conditions (i)-(iii) of Definition 2.1. and such, that it is not able to write new blancs on the tape (i.e. T can overwrite a tape symbol with another tape symbol, but not with a blanc) If c_1 and c_2 are configurations of T and if $c_1 \rightarrow c_2$ then c_2 uses at least the same number of tape cells as c_1. The problem if the machine T halts, after started in the configuration N, is undecidable. Since the machine is deterministic, $\mathcal{O}$(N) is finite iff the machine halts, after started in configuration N.

Once we have a useful T it is easy to construct a useful process $\mathcal{T}$. ■
T and $\mathcal{T}$ will be fixed useful Turing machine and Thue process.

Definition 2.5 *(i) An* $\mathcal{A}$ -tree *(or simply* tree*) is a (not necessarily finite) set* $\mathcal{P}$ *of words over the alphabet* $\mathcal{A}$ *such that* $w \in \mathcal{P}$ *and* $w = uv$ *implies* $u \in \mathcal{P}$ *. Let* $\mathcal{P}_w = \{v : \exists u \in \mathcal{P}\ u = wv\}$. $\mathcal{P}_w$ *is a* (rooted in w) subtree of $\mathcal{P}$.
(ii) Let $\mathcal{W}$ *be a set of words over the alphabet* $\mathcal{A}$ *. By* $\mathsf{pref}(\mathcal{W})$ *we denote the set of all prefixes of words in* $\mathcal{W}$.

Definition 2.6 *(i) Let* $\mathcal{P}$ *be an* $\mathcal{A}$ *-tree. We call a node* $w \in \mathcal{P}$ semiregular *if* $\mathcal{P}_{wv} = \mathcal{P}_{wu}$ *for every production* $\langle v, u \rangle \in \mathcal{T}$.
We call a node w regular *if it is semiregular and if* $i \in \mathcal{P}_w$ *for every static* $i \in \mathcal{A}$.
(ii) Call an $\mathcal{A}$ *-tree* $\mathcal{P}$ semiregular *if every of its nodes is semiregular. Call an* $\mathcal{A}$ *-tree* $\mathcal{P}$ regular *if every of its nodes w is regular or is a leaf.*

The following two lemmas establish a connection between the Thue process and semiregular trees:

Lemma 2.7 *For every set $\mathcal{W}$ of words over $\mathcal{A}$ THUE($\mathcal{W}$) is a semiregular tree.*

Proof: Let $\langle v, u\rangle \in \mathcal{T}$ and let $w \in \mathcal{P} = \mathrm{THUE}(\mathcal{W})$. If $w_1 \in \mathcal{P}_{wu}$ then $wuw_1 \in \mathcal{P}$, hence $wvw_1 \in \mathcal{P}$ and $w_1 \in \mathcal{P}_{wv}$. So $\mathcal{P}_{wu} \subseteq \mathcal{P}_{wv}$. ■

Lemma 2.8 *Every semiregular tree containing a node w contains all elements of the set thue(w)= $\{v : w \overset{*\mathcal{T}}{\Longleftrightarrow} v\}$. So, if there exists a finite semiregular tree containing w then also the sets thue(w) and THUE($\{w\}$) are finite.* ■

Definition 2.9 *(i) Let $\mathcal{H}$ be a Horn clause $Q(\Gamma_1)\wedge Q(\Gamma_2)\wedge\ldots\wedge Q(\Gamma_l) \Rightarrow Q(\Gamma_0)$ where Γ's denote vectors of terms of length equal to the arity of Q. A* $\mathcal{H}$-derivation *is a finite full l-tree $\mathcal{D}$ such that every its node w is labelled with vector $\Delta(w)$ of ground terms of the same length as Γ's and that for each inner node w of $\mathcal{D}$ there exists a substitution σ such that $\sigma\Gamma_0 = \Delta(w)$ and $\sigma\Gamma_i = \Delta(wi)$ for every $i \leq l$.*

(ii) For a set $\mathcal{S}$ of vectors of constant terms an $\mathcal{H}$-derivation will be called a $\mathcal{S}$ -$\mathcal{H}$ -derivation if for every its leaf w $\Delta(w) \in \mathcal{S}$.

Let p be the number of elements of $\mathcal{A}$. We fix a numeration of the symbols of $\mathcal{A}$ with numbers $1, 2 \ldots p$ and identify symbols with their numbers.

Definition 2.10 *$\mathcal{H}$ is the Horn clause $Q(\Gamma_1) \wedge Q(\Gamma_2) \wedge \ldots \wedge Q(\Gamma_p) \Rightarrow Q(\Gamma)$ where Q is an unary relation symbol and $\Gamma = g(\Gamma_1, \Gamma_2, \ldots \Gamma_p)$ is a term built in the language containing exactly one p-ary functional symbol g and no constant symbols. Γ has g in the position w iff w is a word consisting of a single, static symbol of $\mathcal{A}$. The variables in positions u and v in Γ are equal iff $< u, v >\in \mathcal{T}$.*

We assume from now on that terms are built of variables, a functional symbol g and of a constant c. So a constant term is uniquely described by a tree of its symbols g and we identify such a term with this tree. Moreover , we assume that the only predicate symbol in the language is Q, so we can, for simplicity, consider nodes of a derivation as being labelled by (vectors of) ground terms, not by atomic formulas. We put $\mathcal{S} = \{c, g(c, \ldots, c)\}$. Next lemmas describe the structure of a $\mathcal{S}$ -$\mathcal{H}$ -derivation.

Lemma 2.11 *(i) Suppose the root of a $\mathcal{H}$ derivation $\mathcal{D}$ is labelled with a constant term Δ. If there is a node w in the derivation, then $\Delta(w) = \Delta_w$. (where $\Delta(w)$ is, as in Definition 2.9(i), the label of the node w and Δ_w is, as in Definition 2.5(i) a subtree of Δ rooted in w).*

(ii) Suppose the root of a $\mathcal{S}$ -$\mathcal{H}$ -derivation $\mathcal{D}$ is labelled with a constant term Δ . Then Δ is a semiregular tree.

Proof: (i) The proof proceeds by induction on the length of w. Suppose that w and wi are nodes of $\mathcal{D}$, and the label of $\Delta(w)$ is Δ_w. Then, by the definition of $\mathcal{H}$, we have $\Delta(w) = g(t_1, t_2, \ldots t_p)$ and $\Delta(wi) = t_i = \Delta_{wi}$.
(ii) Let w be a node of Δ. We shall prove that w is semiregular. Let v be the longest prefix of w such that $v \in \mathcal{D}$. If $v \neq w$, then v is a leaf of $\mathcal{D}$, and $\Delta(v) \in \mathcal{S}$. By (i) $\Delta(v) = \Delta_v$, so since Δ_w is a subtree of Δ_v w is a semiregular node. If $v = w$ and w is a leaf of $\mathcal{D}$, then w is clearly semiregular. If $v = w$ and w is an inner node, then $\Delta(w)$ is an instance of Γ and so, $\Delta(w)$ is semiregular. Since $\Delta_w = \Delta(w)$, it follows that w is a semiregular node. ■

Lemma 2.12 *(i) Suppose a root of a constant term Δ is a regular node. then there exists a substitution σ such that $\Delta_i = \sigma\Gamma_i$ for $i \in \mathcal{A}$ and $\Delta = \sigma\Gamma$.*
(ii) Suppose a constant term Δ is a regular tree. Then Δ is a label of the root of a $\mathcal{S}-\mathcal{H}$-derivation $\mathcal{D}$. $\mathcal{D}$ is a full p-tree that has the same set of inner nodes as Δ. If w is a leaf of Δ then it is labelled by $g(c, c, \ldots, c)$ in $\mathcal{D}$. If w is a leaf of $\mathcal{D}$ but is not in Δ then it is labelled by c in $\mathcal{D}$.

Proof: (i) It immediately follows from the definition of $\mathcal{H}$ that if $\Delta = \sigma\Gamma$ then also $\Delta_i = \sigma\Gamma_i$, for $i \in \{1, 2, ..., p\}$. So, to prove that there exists a substitution σ, such that (i) holds, it suffices to check that
(iii) if x is a variable which occurs in position v in Γ, then position v exists in Δ, and
(iv) if a variable x occurs in Γ in positions v and u then $\Delta_v = \Delta_u$.

To prove (iii), notice that if Γ has a variable in position v, then $v = wj$ and, by regularity, the term Δ has the function symbol g in position w, so it has some symbol (either g or c) in position wj. To prove (iv) suppose that Γ has variable x in positions v and u. Then $< v, u > \in \mathcal{T}$. By semiregularity of the root, if $u \in \Delta$, then $\Delta_u = \Delta_v$. If $u \notin \Delta$ then by (iii), Δ has the constant symbol c in positions v and u. So, the existence of a substitution σ satisfying the conditions of (i) is established.
(ii) Assume that Δ is a regular tree, and let $\mathcal{D}$ be the least full p-tree containing all regular nodes of Δ as inner nodes. Let the nodes w of $\mathcal{D}$ which are in Δ be labelled by Δ_w, and let the other nodes be labelled by the term c. It is easy to prove that such a labeling defines a valid $\mathcal{S}$ -$\mathcal{H}$ -derivation. ■

2.2 From Semiregular to Regular Trees

In [19] we prove, that every finite semiregular tree can be extended to a finite regular tree. Now, we shall examine the construction of the extension more carefuly.

Consider the finite tree THUE({ N }) , where N is as in Definition 2.1. (N is a configuration of the Turing machine, identified with a word over $\mathcal{A}$). Let d denote the depth of this tree. Put $\mathcal{P}_d$= THUE({ N }) , and for given $\mathcal{P}_k$ $(k = d, d-1, d-2, \ldots 1)$ let $\mathcal{P}_{k-1}$ be the result of the following

Procedure: - put $\mathcal{P}$:=$\mathcal{P}_k$.
-**while** there exists a nonregular inner node w of $\mathcal{P}$, such that $|w| = k$, **do** take any static symbol i such, that $wi \notin \mathcal{P}$ and put $\mathcal{P} := THUE(\mathcal{P} \cup \{wi\})$.
-put $\mathcal{P}_{k-1} := \mathcal{P}$.

Lemma 2.13 *(i) If w is an inner node of $\mathcal{P}_d$ then w contains no dynamic symbols.*
(ii) If w is an inner node of $\mathcal{P}_k$ $(k \leq d)$ then w contains no dynamic symbols.
(iii) The tree THUE($\mathcal{P} \cup \{wi\}$) in the substitution above is finite.
(iv) The procedure halts.
(v) If $w \in \mathcal{P}_0$ and $|w| \geq k$ then $w \in \mathcal{P}_k$.
(vi) $\mathcal{P}_0$ is a regular tree.
(vii) If $w \in \mathcal{P}_{n+1}$ and $|w| \geq n$ then the n'th symbol of w is a dollar.
(viii) If $w \in \mathcal{P}_n$ and the n'th symbol of w is not a dollar then $|w| = n$.

Proof: (i) It follows, since N has only one dynamic symbol, as its last symbol and since there is no production in $\mathcal{T}$ that can write a static symbol to the right of it.

(ii) The procedure chooses w that is not a leaf of $\mathcal{P}$ and hence contains no dynamic symbols. i is a static symbol. There is no production in $\mathcal{T}$ that can increase the number of dynamic symbols in a word. So, every $v \in \mathrm{THUE}(wi)$ contains only static symbols. $THUE(\mathcal{P} \cup \{wi\}) = THUE(\mathcal{P}) \cup THUE(wi)$. Since $\mathcal{P}$ is already semiregular the right hand side of the last equality equals $\mathcal{P} \cup THUE(wi)$.
(iii) $\mathrm{THUE}(wi)$ contains only words not longer, than $|wi|$.
(iv) The procedure outputs a sequence of trees increasingly ordered by inclusion. None of them is deeper than d.
(v) That follows from (ii) and (iii).
(vi) If $w \in \mathcal{P}_0$ and $|w| = k$ then w becames to be a regular node in $\mathcal{P}_k$ and it cannot change later.
(vii) $w \in \mathrm{THUE}(v)$ then, where v has a dollar as its n'th symbol. Using $\mathcal{T}$ cannot change it.
(viii) By (vii), $w \notin \mathcal{P}_{n+1}$. So, $w \in \mathrm{THUE}(v)$ for some v of lenght n and not containing any dynamic symbol. Hence $|w| = n$. ■

Let q_0 be the initial state of the Turing machine T. Let $a = \langle !, 0, 0\rangle$, $b = \langle 0, 0, 0\rangle$ and $c = \langle \$, q_0, 1\rangle$. So, we identify the configuration N of the T with the tree of prefixes of the word $ab^{n-2}c$. We shall now prove that, while $\mathcal{P}_d$ is an uncontrolable tree, the intersection $\mathcal{P}_0 \cap \{a, b, c\}^*$ is an easily computable function of n.

Lemma 2.14 *(i) if $w \in \mathcal{P}_0$ and w contains a, then a is the first or the last symbol of w.*
(ii) if b is the first symbol of some $w \in \mathcal{P}_0$, then $w = b$.
(iii) if $w \in \mathcal{P}_0$ has c as its last symbol, then $w \in \mathcal{P}_d$.

Proof: (i) It is so in $\mathcal{P}_d$. The **Procedure** in the process of constructing $\mathcal{P}_k$ can add the static symbol a to the tree as the last $k+1$'th symbol of some word. No symbols are added to the tree in a position deeper than $k+1$ after that, so the a will remain the last symbol of every word that contains it.
(ii) A word w that has b as its first symbol can be added only during the construction of $\mathcal{P}_0$ from $\mathcal{P}_1$. Only words of length 1 are added then.
(iii) c is a dynamic symbol. $\mathcal{P}_0$ has such symbols on the same positions as $\mathcal{P}_k$. ■

Definition 2.15 *Let* $\mathrm{N}_0 = \{ab^k : k \leq n-1\} \cup \{ab^k a : k \leq n-2\} \cup \{b, ab^{n-2}c\}$.

Lemma 2.16 $\mathcal{P}_0 \cap \{a, b, c\}^* = \mathrm{N}_0$.

Proof: To prove that the inclusion $\subseteq$ holds notice that, by Lemma 2.13 (vi) $\mathcal{P}_0$ is a regular tree. Since $\mathrm{N} \subseteq \mathcal{P}_0$ all inner nodes of N are regular nodes of $\mathcal{P}_0$. If $k \leq n-2$ then ab^k is an inner node of N. So, if $k \leq n-2$ then $ab^k a \in \mathcal{P}_0$ and $ab^{k+1} \in \mathcal{P}_0$. Empty word is an inner node of N so it is regular in $\mathcal{P}_0$. Hence $b \in \mathcal{P}_0$.

For the $\supseteq$ inclusion, by the Lemma 2.14.(iii) we must prove that

$$\mathcal{P}_0 \cap \{a, b\}^* = \{ab^k : k \leq n-1\} \cup \{ab^k a : k \leq n-2\} \cup \{b\}$$

Since neither a nor b is a dollar, by Lemma 2.13 (v),(vii) the depth of $\mathcal{P}_0 \cap \{a, b\}^*$ is not greater than n. By Lemma 2.14 (ii) if $w \in \mathcal{P}_0 \cap \{a, b\}^*$ and $w \neq b$ then the first symbol of w is a. By Lemma 2.14. (i) w is of the form ab^k or $ab^k a$. ■

As a summary of all the results above we get:

Lemma 2.17 (Main Lemma) *The problem*

Given n, does there exist a $\mathcal{S} - \mathcal{H}$-derivation $\mathcal{D}$ such that $\{a, b, c\}^* \cap \{$the set of inner nodes of $\mathcal{D}$ and its leaves labelled with $g(c, c, \ldots, c)\} = \mathrm{N}_0$?

is undecidable.

Proof: If $\mathcal{D}$ exists then, by Lemma 2.11 its root is labelled by a finite semiregular tree Δ such that $\mathrm{N}\subseteq\ \Delta$. If there exists a finite semiregular tree Δ such that $\mathrm{N}\subseteq\ \Delta$ then, by Lemmas 2.13 and 2.16. there exists a regular tree $\mathcal{P}_0$ such that $\mathcal{P}_0 \cap \{a,b,c\}^* = \mathrm{N}_0$. By Lemma 2.12 (ii) there exists a derivation $\mathcal{D}$ then. So the problem of the lemma is equivalent to the one: *Given n. Does there exist a semiregular finite tree Δ such that* $\mathrm{N}\subseteq\ \Delta$. The last is undecidable by Definition 2.1 (iv). ■

2.3 The Main Theorem

Definition 2.18 *$\mathcal{H}^\star$ is the Horn clause:*
$Q_1(\Gamma_1, x_1) \wedge Q_1(\Gamma_2, x_2) \wedge \ldots \wedge Q_1(\Gamma_p, x_p) \Longrightarrow Q_1(\Gamma, g(y_1, y_2, \ldots y_p))$
where Q_1 is a binary relation symbol, $\Gamma = g(\Gamma_1, \Gamma_2, \ldots \Gamma_p)$ are as in clause $\mathcal{H}$ and y's and x's are variables not occuring in Γ's. If $i \in \{a,b,c\}$ then x_i is the same variable as y_i . Variables not defined as equal are distinct.

The nodes of $\mathcal{H}^\star$-derivation are labelled with pairs of terms. So, the legal premises of the derivation are also pairs of terms:

Definition 2.19 $\mathcal{S}^\star = \{\langle c, c\rangle, \langle g(c, c, \ldots, c), g(c, c, \ldots c)\rangle\}$.

Lemma 2.20 *Let w be a node of a $\mathcal{S}^\star - \mathcal{H}^\star$ -derivation $\mathcal{D}$. If $\langle \Delta^1, \Delta^2\rangle$ is the label of w then $\Delta^2 = \Delta^1 \cap \{a,b,c\}^*$.*

Proof: Induction on the depth of the tree $\mathcal{D}_w$. ■
Now, from the main lemma we get

Lemma 2.21 *The problem*
Given n, does there exist a $\mathcal{S}^\star - \mathcal{H}^\star$ -derivation with its root labelled with $\langle x,$ $\mathrm{N}_0\rangle$, where x is any term (we know that it is $\mathcal{P}_0$) ?
is undecidable. ■

We still have an unpredictable term in the root of the derivation. We shall do a trick to hide it:

Definition 2.22 *$\mathcal{H}^{\star\star}$ is the Horn clause*
$Q(v_1, z, v_2) \wedge Q(\Gamma_1, x_1, z_1) \wedge Q(\Gamma_2, x_2, z_2) \wedge \ldots \wedge Q(\Gamma_p, x_p, z_p) \Rightarrow Q(\Gamma, g(y_1, y_2, \ldots y_p), z)$
where Q is a ternary relation symbol , Γ's , x's and y's are as in clause $\mathcal{H}^\star$, z's and v's are new distinct variables .

Definition 2.23 $\mathcal{S}^{\star\star} = \{\langle c, c, c\rangle, \langle g(c, c, \ldots, c), g(c, c, \ldots c), c\rangle\}$.

It is easy to observe, that the following lemma holds:

Lemma 2.24 *There exist a $\mathcal{S}^\star - \mathcal{H}^\star$ -derivation with its root labelled with $\langle x,$ $\mathrm{N}_0\rangle$, where x is any term, if and only if there exists a $\mathcal{S}^{\star\star} - \mathcal{H}^{\star\star}$-derivation with its root labelled with $\langle \sigma\Gamma, g(c, c, \ldots, c),$ $\mathrm{N}_0\rangle$, where σ substitutes c for every variable.* ■

Now, from the last lemma and from Lemma 2.21:

Theorem 2.25 (The Main Theorem) *The problem*
Given n, does there exist a $\mathcal{S}^{\star\star} - \mathcal{H}^{\star\star}$-derivation with its root labelled with $\langle \sigma\Gamma, g(c, c, \ldots, c),$ $\mathrm{N}_0\rangle$, where σ substitutes c for every variable?
is undecidable. ■

3 The Linial-Post Theorem

We will use a version of the theorem about undecidability of the Cycle Unification Problem a bit stronger than the one presented in [7] and[23]. The stronger result can be obtained by a refinement of the argument in[23] but we shall not present the argument here since we are not going to report all the proof.

Lemma 3.1 *For the first order language without constant symbols and with only one binary function symbol f the problem*
given a Horn clause $\mathcal{H} = (Q(t_1) \Longrightarrow Q(t_2))$ and terms s_1 and s_2, is there any substitution σ such that $Q(\sigma(s_2))$ is a logical consequence of $\mathcal{H}$ and of $Q(\sigma(s_1))$?

is undecidable. ■

When the substitution in the Lemma 3.1. exists then we say that the answer for Cycle Unification Problem for data t_1, t_2, s_1 and s_2 is positive.

As in Section 1.3. consider the language of terms of propositional calculus built with variables and the symbol $\Longrightarrow$. We shall prove the following (a strong version of the Linial- Post theorem in Section 1.3.)

Theorem 3.2 *Let $\mathfrak{l}$ be a tautology that is not of the form $p \Longrightarrow p$ for some term p. Then the problem*
Given a set of tautologies of propositional calculus $\mathcal{S}$. Does $\mathcal{S} \vdash \mathfrak{l}$
is undecidable

Proof: We shall prove the theorem giving the reduction of the Cycle Unification Problem to the problem of the last theorem.

Similarly (but not in quite the same way) as in section 1.2. let δ be the most general substitution such that $\delta(t_1) = \delta(s_1)$. Let $\mathcal{H}_1 = \delta(\mathcal{H})$ and suppose $\mathcal{H}_k$ $=(Q(r_1^k) \Longrightarrow Q(r_2^k))$ (we assume that the variables in r's are distinct from those in s's and t's). Let σ be the most general substitution such that $\sigma(t_1) = \sigma(r_2^k)$. Then define $\mathcal{H}_{k+1} = Q(\sigma(r_1^k) \Longrightarrow \sigma(t_2))$.

Theorem 3.1. is equivalent to the undecidability of the following problem (with the assumptions of Theorem 3.1.): *Given a unary Horn clause $\mathcal{H} = (Q(t_1) \Longrightarrow Q(t_2))$ and two terms s_1 and s_2. Does there exist a natural number n such that: $\mathcal{H}_n$ $=(Q(r_1^n) \Longrightarrow Q(r_2^n))$, and r_2^n is an instance of s_2* is undecidable.

Now we can construct the reduction. The first idea is, given the data of Theorem 3.1. put $\mathcal{S} = \{\mathcal{H}, s_1, f(s_2,\mathfrak{l})\}$, where we do not see any difference between the (binary functional) symbols $\Longrightarrow$ and f in $\mathcal{H}$. We claim that if the answer for the Cycle Unification Problem for data t_1, t_2, s_1 and s_2 is positive then $\mathcal{S} \vdash \mathfrak{l}$. In fact, let n be a natural number such that r_2^n is an instance of s_2. Take n copies of $\mathcal{H}$ with pairwise disjoint sets of variables and the most general substitution σ^*, such that $\sigma^*(t_2^k) = \sigma^*(t_1^{k+1})$ for $1 \leq k \leq n$ (where the k'th copy of $\mathcal{H}$ is $Q(t_1^k) \Longrightarrow Q(t_2^k)$), $\sigma^*(t_1^1) = \sigma^*(s_1)$ and $\sigma^*(t_2^n) = \sigma^*(s_2)$ (σ^* is something like superposition of δ and all the σ's in the remarks above).

$\sigma^*(\mathcal{H})$ is an instance of $\mathcal{H}$ for every copy of $\mathcal{H}$ and hence is a provable term. The left hand side of the σ^* of the first copy $Q(t_1^1) \Longrightarrow Q(t_2^1)$ of $\mathcal{H}$ is an instance of s_1, which is a provable term , so also the right hand side of σ^* of the first copy of $\mathcal{H}$ is a provable term. By induction, also t_2^k is a provable term. Now we can use a proper instance of $f(s_2,\mathfrak{l})$ to derive $\mathfrak{l}$.

The above method is an easy one but, unfortunately, not quite correct. It is, in general not true that if $\mathcal{S} \vdash \mathfrak{l}$ then $\mathcal{H}$ and some instance of $Q(s_1)$ imply some instance of $Q(s_2)$. To overcome the problem we first require a bit more of $\mathcal{H}$, s_1 and s_2:

Definition 3.3 *(i) t is a* symmetric *term if $t = f(s, s)$ for some term s.*
(ii) A term t is asymmetric *if there is no such substitution ς that $\varsigma(t)$ is symmetric.*
(iii) $\bar{t_1} = f(f(t_1, f(v, v)), f(t_1, f(v, v)))$, $\bar{t_2} = f(f(t_2, v), f(t_2, v))$,
$\bar{s_1} = f(f(s_1, y), f(s_1, y))$, $\bar{s_2} = f(f(s_2,ł), f(s_2,ł))$, *$v, y$ are new variables here.*
We suppose that the variables in ł are distinct from those in the other terms.

We put $\bar{\mathcal{S}} = \{\bar{\mathcal{H}}, \bar{s_1}, f(\bar{s_2},ł)\}$, where $\bar{\mathcal{H}} = (\bar{t_1} \Longrightarrow \bar{t_2})$. The following easy lemma allows us to reduce the Cycle Unification Problem to a case of data satisfying some conditions of symmetry:

Lemma 3.4 *(i) $\bar{t_1}, \bar{t_2}, \bar{s_1}, \bar{s_2}$ are symmetric terms.*
(ii) $f(\bar{t_1}, \bar{t_2})$ in an asymmetric term.
(iii) $f(\bar{s_2},ł)$ is an asymmetric term.
(iv) The answer for Cycle Unification Problem for data $\bar{t_1}, \bar{t_2}, \bar{s_1}, \bar{s_2}$ is the same as for data t_1, t_2, s_1, s_2. If the answer is positive and σ^*, is a most general substitution such that $\sigma^*(t_2^k) = \sigma^*(t_1^{k+1})$ for $1 \leq k \leq n$ (where the k'th copy of $\mathcal{H}$ is $Q(t_1^k) \Longrightarrow Q(t_2^k)$), $\sigma^*(t_1^1) = \sigma^*(s_1)$ and $\sigma^*(t_2^n) = \sigma^*(s_2)$ then there exists a substitution $\bar{\sigma}^*$ such that $\bar{\sigma^*}(\bar{t_2^k}) = \bar{\sigma^*}(\bar{t_1^{k+1}})$, $\bar{\sigma^*}(\bar{t_1^1}) = \bar{\sigma^*}(\bar{s_1})$ and $\bar{\sigma^*}(\bar{t_2^n}) = \bar{\sigma^*}(\bar{s_2})$. Moreover $\bar{\sigma^*}(ł) = ł$.
(v) All the terms in $\bar{\mathcal{S}}$ are tautologies.
(vi) Every $\bar{\sigma^*}(\bar{t_1^k})$ is a symmetric term. ■

We claim that the answer for Cycle Unification Problem for $\bar{t_1}, \bar{t_2}, \bar{s_1}, \bar{s_2}$ is negative then the only new terms that can be derived from $\bar{\mathcal{S}}$ are instances of some $\bar{\sigma^*}(\bar{t_1^k})$. Suppose, that t is not an instance of some $\bar{\sigma^*}(\bar{t_1^k})$ and that $\bar{\mathcal{S}} \vdash t$. We suppose, that t is the first such term, in the meaning , that t can be derived from $\bar{\mathcal{S}}$ and some instances of some $\bar{\sigma^*}(\bar{t_1^{k+1}})$ by a single use of modus ponens. Only two of those terms: $\bar{\mathcal{H}}$ and $f(\bar{s_2},ł)$ can be productive as modus ponens major premise. The remaining terms are symmetric. $f(\bar{s_2},ł)$ can not be used as modus ponens major premise since, by the hypothesis, neither $\bar{s_1}$ nor any of instances of some $\bar{\sigma^*}(\bar{t_1^k})$ can be unified with $\bar{s_2}$ and since $\bar{\mathcal{H}}$, because of its asymmetry, can not be unified with symmetric $\bar{s_2}$. So we can use only $\bar{\mathcal{H}}$ as the modus ponens major premise. Now, what can be used as modus ponens minor premise: By the symmetry argument $\bar{\mathcal{H}}$ can not be used, since it is asymmetric and its left hand side is symmetric. If $\bar{s_1}$ or any of instances of some $\bar{\sigma^*}(\bar{t_1^k})$ is used then we get another instance of some $\bar{\sigma^*}(\bar{t_1^k})$ as a conclusion. $f(s_2,ł)$ can not be used as modus ponens minor premise since it is asymmetric. So, if the answer for Cycle Unification Problem for $\bar{t_1}, \bar{t_2}, \bar{s_1}, \bar{s_2}$ is negative then only symmetric new terms can be derived from $\bar{\mathcal{S}}$. Since ł is not symmetric it is not the case that $\bar{\mathcal{S}} \vdash ł$. This ends the proof of Theorem 3.2. ■

If we choose ł to be such a tautology that gives full axiomatization of propositional calculus we get

Theorem 3.5 *The problem*
Given a set $\mathcal{S}$ of tautologies of propositional calculus. Is $\mathcal{S}$ an axiomatization of propositional calculus.
is undecidable.

4 Acknowledgement

I would like to thank my adviser and co-author of [19] Leszek Pacholski, who has shown me a nice rock (see Section 2.) and ordered me to climb on it.

5 References

[1] K. Apt: Logic Programming; in Handbook of Theoretical Computer Science vol B., J. van Leeuwen (ed). 1990.
[2] C. Beeri, M.Y. Vardi: the implication problem for data dependencies; Proc. International Coll. on Automata, Languages and Programming 1981, Lecture Notes in Computer Science, vol 115, pp 73-85, Springer, Berlin.
[3] W. Bibel, S. Hölldobler and J. Würtz: Cycle Unification; in D. Kapur (ed.), Proc. the Conference on Automated Deduction, Lecture Notes in Artificial Intelligence, vol 607, Springer (June 1992).
[4] A. K. Chandra, M. Y. Vardi: The implication problem for functional and inclusion dependencies is undecidable; SIAM J. Comput. vol 14, No 3, (1985).
[5] C.L. Chang, R.C.T. Lee: Symbolic Logic and Mechanical Theorem Proving; Academic Press, New York, San Francisco, London 1973
[6] P. Devienne, P. Lebégue and J.C. Routier: Halting Problem of one Binary Horn Clause is undecidable; proc. of STACS 93, Springer, LNCS, vol 665 pp 48-58.
[7] P.Devienne, P.Lebégue J.C.Routier:Cycle Unification is Undecidable. prepr. 1992
[8] G.Gottlob, A.Leitsch:Fast Subsumption Algorithms; LNCS 204-II, Springer, 1985.
[9] G. Gottlob, A. Leitsch: On the Efficiency of Subsumption Algorithms; JACM vol 32 No. 2 (April 1985), pp. 280-295.
[10] G. Gottlob, A. Leitsch: Deciding Horn clause implication problem by ordered semantic resolution; Comp. Intelligence II, F. Gardin, G. Mauri, (ed), 1990.
[11] Y.S. Gurevich: The problem of equality of words for certain classes of semigroups; (Russian), Algebra i Logika 5,(1966) pp 25-35
[12] P. Hanschke, J. Würtz: Satisfiability of the Smallest Binary Program; Information Processing Letters 496 (1993) pp. 237-241.
[13] C. Herrmann: On the undecidability of implications between embedded multivalued database dependencies; preprint.
[14] P.C. Kanellakis: Elements of Relational Database Theory; in Handbook of Theoretical Computer Science vol B., J. van Leeuwen (ed). 1990.
[15] A. Leitsch, Implication Algorithms for Classes of Horn Clauses; Statistik, Informatik und Ökonomie, Springer Verlag 1988, pp 172-179.
[16] A. Leitsch, Deciding Horn Clauses by Hyperresolution, Proc. Computer Science Logic 89, Lecture Notes in Computer Science 440 (1989), pp 225-241
[17] S. Linial, E. Post: Recursive unsolvability of the deducibility, Tarski's completeness and independence of axioms problems of the propositional calculus (abstract); Bulletin of the American Mathematical Society, 55 p. 50 (1949).
[18] R. McKenzie: On Spectra, and the Negative Solution of the Decision Problem for Identities Having a Finite Nontrivial Model; The Journal of Symbolic Logic, vol 40, No 2 pp 186-195, 1975.
[19] J. Marcinkowski, L. Pacholski: Undecidability of the Horn Clause Implication Problem; Proc 33rd Annual Symposium on the Foundations of Computer Science (1992). pp. 354-362.

[20] W.E. Singletary: Results Regarding the Axiomatization of Partial Propositional Calculi; Notre Dame Journal of Formal Logic, vol IX, Number 3, 1968.
[21] R.B.Stillman: The Concept of Weak Substitution In Theorem-proving; JACM vol.20.4(Oct. 1973), 648-667.
[22] M.Schmidt-Schauss: Implication of Clauses is Undecidable; Theoretical Computer Science 59 (1988), pp. 287-296.
[23] J. Würtz: Unifying Cycles; ECAI 1982, B. Neumann ed. pp 60-64.
[24] M. K. Yntema: A detailed argument for the Post-Linial theorems; Notre Dame Journal of Formal Logic, vol V, (1964). pp 35-51.