Total Homeostaticity and Integrity Constraints Restorability Recognition [1]

Michael I. Dekhtyar
Dept. of CS, Tver St. Univ.
3 Zheljabova str. Tver, Russia, 170000
dekhtyar@tversu.ac.ru

Alexander Ja. Dikovsky
Keldysh Inst. for Appl. Math.
4 Miusskaya sq. Moscow, Russia, 125047
dikovsky@spp.keldysh.ru

Abstract

We introduce and explore a property of deductive data bases with updates wich we call total homeostaticity, and which substantially generalizes the following property: "for any given external update violating the integrity constraints there exists a reaction of the data base, which restores the integrity constraints". This property is characteristic of active data bases. We explore the computational complexity of total homeostaticity recognition in various subclasses of Datalog programs with updates and show that it is much simpler than the homeostaticity in a particular DB state. It turns out to be polynomial time solvable in several situations of interest for applications.

1 Introduction

In this work the deductive data bases (*DDBs*) serve as a model of interactive discrete dynamic systems with complex states described by relations over values of multiple parameters. At any given moment the system state is represented by a data base state (*DB state*), i.e. by a finite set of extensional facts. Possible *actions* of the system (i.e. transitions from one state to another) are described by a logic program with updates $\mathcal{P}$ over DB states. More specifically, a transition from state $\mathcal{E}_1$ to state $\mathcal{E}_2$ is described as a computation of some predefined goal $:- a$ of $\mathcal{P}$ transforming the first DB state into the second: $\mathcal{E}_1 \stackrel{a}{\vdash_{\mathcal{P}}} \mathcal{E}_2$. External updates are considered as *disturbances* of an active medium of the system, and are represented by binary relations $\stackrel{d}{\longrightarrow}$ on DB states. One can always evaluate the effect of a committed disturbance on the current state. However, *one cannot predict in*

[1] This work was sponsored by INTAS (Grant 94-2412) and by the Russian Fundamental Studies Foundation (Grant 96-01-00395).

a given state which disturbance is to be committed. E.g., in a bank there exists a list of feasible services required by its clients (external updates), and the corresponding transactions. Transactions are predictable, therefore in this example they correspond to actions of the system. On the other hand, one cannot predict exactly which orders for the services emerge at the given moment. Therefore, they correspond to disturbances of an active medium of the system.

Such interactive approach to external updates is aimed at the analysis of the global behavior of a discrete dynamic system in the space of its states. The local behavior of the system in a current state $\mathcal{E}_0$ is described as one interaction with its medium in this state. There are two types of interactions depending on what comes first, a medium disturbance or a system action: $\mathcal{E}_0 \stackrel{d}{\longrightarrow} \mathcal{E}_1^* \stackrel{a}{\vdash_{\mathcal{P}}} \mathcal{E}_1$ and $\mathcal{E}_0 \stackrel{a}{\vdash_{\mathcal{P}}} \mathcal{E}_1^* \stackrel{d}{\longrightarrow} \mathcal{E}_1$. Normally, one can distinguish between feasible and not feasible interactions, depending on a criterion of admissibility of system states. Each feasible interaction applies to an admissible state $\mathcal{E}_0$ and yields an admissible state $\mathcal{E}_1$. However, the intermediate state $\mathcal{E}_1^*$ may in general be inadmissible, in which case the reaction compensates for the ruinous stimuli. The admissibility criterion is considered as *integrity constraints (IC)* and is expressed by a formula or a (logic) program Φ over DB states. In terms of the IC the feasibility of the interaction is expressed as follows: the interaction of the form $\mathcal{E}_0 \stackrel{d}{\longrightarrow} \mathcal{E}_1^* \stackrel{a}{\vdash_{\mathcal{P}}} \mathcal{E}_1$ or $\mathcal{E}_0 \stackrel{a}{\vdash_{\mathcal{P}}} \mathcal{E}_1^* \stackrel{d}{\longrightarrow} \mathcal{E}_1$ is *feasible* if $\mathcal{E}_0 \models \Phi$ and $\mathcal{E}_1 \models \Phi$. So a discrete dynamic system is represented by a *deductive data base (DDB)* $\mathcal{B} =< \mathcal{P} \cup GOALS,\ \Phi >$, and its medium is described by a relation $\stackrel{d}{\longrightarrow}$ on DB states of $\mathcal{B}$.

The global behavior of the system in a state $\mathcal{E}_0$ is represented by sequences of (feasible) interactions starting in $\mathcal{E}_0$: the *(feasible) trajectories.* Basically, we have two types of trajectories corresponding to the two types of interactions above: *disturbance-action* trajectories

$$\mathcal{E}_0 \stackrel{d_1}{\longrightarrow} \mathcal{E}_1^* \stackrel{a_1}{\vdash} \mathcal{E}_1 \stackrel{d_2}{\longrightarrow} \mathcal{E}_2^* \stackrel{a_2}{\vdash} \mathcal{E}_2 \ ...$$

and *action-disturbance* trajectories

$$\mathcal{E}_0 \stackrel{a_1}{\vdash} \mathcal{E}_1^* \stackrel{d_1}{\longrightarrow} \mathcal{E}_1 \stackrel{a_2}{\vdash} \mathcal{E}_2^* \stackrel{d_2}{\longrightarrow} \mathcal{E}_2 \ ...$$

Infinite feasible disturbance-action trajectories represent a *homeostatic* behavior of the DDB: the DDB always manages to restore the IC by its actions along this trajectory in spite of disturbances of its medium which violate the IC. Therefore, such trajectories are called *homeostatic.* Dually, the infinite feasible action-disturbance trajectories represent a *stable* behavior of the DDB: the disturbances of the medium along the trajectory always compensate for possible ruinous actions of the DDB. Therefore, such trajectories are called *stable.*

Trajectories of both types form (infinite) trees with the root $\mathcal{E}_0$: $T_{da}(\mathcal{E}_0)$ and $T_{ad}(\mathcal{E}_0)$ respectively. A number of natural properties of interactive behavior of $\mathcal{B}$ in a given DB state can be formalized in terms of these trees.

Definition 1 *Let* $\mathcal{B}$ *be a DDB and* $\stackrel{d}{\longrightarrow}$ *be a disturbance relation. Let* $Q_1, Q_2 \in \{\forall, \exists\}$. *Then* $\mathcal{B}$ *is* $d - Q_1Q_2$-homeostatic in DB state $\mathcal{E}_0$ *if in the tree* $T_{da}(\mathcal{E}_0)$ *there is a* Q_1Q_2*-subtree in which all branches are infinite homeostatic trajectories.* $\mathcal{B}$ *is* $d - Q_1Q_2$-stable in DB state $\mathcal{E}_0$ *if in the tree* $T_{ad}(\mathcal{E}_0)$ *there is a* Q_1Q_2-*subtree in which all branches are infinite stable trajectories.*

In [3] we introduce the $\exists\exists$-stability and explore its computational complexity. In [4] the $\forall\exists$-homeostaticity is introduced and investigated. Some other types of homeostatic and stable behavior of a DDB in a specific DB state are studied in [5].

This paper is devoted to the property of *total homeostaticity* of a DDB, i.e. homeostaticity in each DB state:

$$\forall \mathcal{E}, \mathcal{E}_1^* \ (\mathcal{E} \models \Phi \ \& \ \mathcal{E} \stackrel{d}{\longrightarrow} \mathcal{E}_1^* \Rightarrow \exists a, \mathcal{E}_1 \ (\mathcal{E}_1^* \stackrel{a}{\vdash} \mathcal{E}_1 \ \& \ \mathcal{E}_1 \models \Phi)).$$

It is equivalent to the $\forall\exists$-homeostaticity in any admissible DB state. This property has attracted a widespread attention because it corresponds to an important and long known scenario in DDBs applications. In order for a DB state of a propositional DDB, violating the IC, to be transformed into a DB state satisfying the IC, the so called revision algorithms (see e.g. [10]) are applied. In fact the same kind of transformations is investigated in [11, 12] in a more general context. In [11, 12] the IC is expressed by a rather general revision program $\mathcal{P}$ and partial algorithms are proposed transforming a given inadmissible DB state $\mathcal{E}$ into a stable model of $\mathcal{P}$ (in the sense of [7]), "induced" by $\mathcal{E}$. In both cases external updates are treated implicitly: the initial inadmissible DB state may be viewed as the result of such a ruinous update. In [9] external elementary updates (insertion or deletion of a ground literal) are explicit, the IC Φ is expressed by simple productions (rules), and algorithms are described "constructing" a model of Φ consistent with an input update. So in these papers external DB updates correspond to destructive disturbances and the algorithms restoring the IC correspond to the restoring DDB actions in our terms. This is in fact the basic scenario in active data bases (cf. [2, 8]), where the disturbances are either elementary updates or some external events recognizable by action rules. Here again the total homeostaticity is guaranteed by some strategy over actions.

It should be stressed that while in the cited literature one is always interested in methods which **guarantee** the total homeostaticity, we are investigating the inverse problem: *given a DDB* $\mathcal{B}$ *in a class of DDBs* $\mathbf{P}$, *an IC* Φ *in a class of ICs* $\mathbf{I}$, *and a disturbance relation* $\stackrel{d}{\longrightarrow}$ *which is restricted in some sense, one should* **recognize** *whether* $\mathcal{B}$ *is totally homeostatic with respect to* $\stackrel{d}{\longrightarrow}$.

The data base scenario above is only one among many others, where the property of total homeostaticity describes the intended steady behavior of a discrete system in an active medium. This property is peculiar to various interactive systems whose operation involves consumption, compensation, and

restoration of some resources. The typical examples are plants, trade enterprises, warehouses, etc. Their medium is the source of appearing orders, and their activity is aimed at filling all of them and in so doing, supporting their successful and unlimited operation. Another classical example are the homeostatic biological systems.

In this paper we explore the computational complexity of the property of total $\forall\exists$-homeostaticity of DDBs. Not surprisingly, this property is undecidable in the class of all DDBs. However, it is solvable in the class of DDBs whose logic programs are in $Datalog^u$ (i.e. Datalog extended by elementary updates). We prove in fact that in this class the total homeostaticity is $TIME(2^{2^{poly}})$-complete. Imposing various natural constraints on recursion, clauses, updates, and ICs, we give the corresponding strict complexity bounds which turn out to be rather tractable in some classes of DDBs of interest for applications. E.g., we prove that in the case where $Datalog^u$-programs are stratified with respect to the elementary updates, and the arity of intensional predicates is bounded (or when this is not the case, the programs are ground) this problem is $PSPACE$-complete. If in addition we assume that logic programs are branching (non recursive) and the intensional signature is fixed then the problem is Π_2^P-complete (even co-NP-complete when the programs are ground). Moreover, the total homeostaticity is polynomial time solvable if the IC is monotonic and the logic programs are ground, positive (i.e. do not use negation), and either branching with fixed intensional signature or are production systems. It is interesting to note that the total homeostaticity is considerably simpler than the homeostaticity in a particular DB state. E.g., as we have shown in [4], the homeostaticity in a particular DB state with respect to polynomial time ICs is $TIME(2^{poly})$-complete even in the class of ground positive production systems.

Although sometimes the complexity of this problem appears to be too high, there are natural methods of factorization of the space of all DB states (discussed in section 5), which can substantially decrease the size of the problems in practical DDB applications.

2 Basic Notation and Definitions

We consider a 1st order language $\mathbf{L}$ in a signature containing: pairwise disjoint sets $\mathbf{P}^e$ of *extensional predicates*, $\mathbf{P}^q$ of *intensional query predicates,* and $\mathbf{P}^u$ of *intensional update predicates,* and a set of function (and constant) symbols $\mathbf{F}$. $\mathbf{H}$ denotes the Herbrand universe of $\mathbf{L}$, and $\mathbf{B}^e$, $\mathbf{B}^i$ denote the extensional and the intensional Herbrand bases respectively. *DB states* are finite subsets of $\mathbf{B}^e$. We consider *logic programs with updates* in $\mathbf{L}$ with clauses of the form $Head \coloneq Body$, where $Head$ is an intensional atom $p(\bar{t})$ with $p \in \mathbf{P}^q \cup \mathbf{P}^u$, and $Body$ is a (possibly empty) sequence of:

- literals of the form $q(\bar{u})$ or $\neg r(\bar{v})$ with $r \notin \mathbf{P}^u$, (i.e. only atoms with extensional and intensional query predicates, can be negated),

- elementary DB updates $insert(Fact)$, $delete(Fact)$, where $Fact$ is an extensional atom,

- assignments of the form $X := e$ and arithmetical constraints of the form $e_1 < e_2$ with e, e_1, e_2 being arithmetical expressions.

So we subsume that numerals and arithmetic operators are included into **F**. The semantic scheme below will show that they are interpreted in the standard way.

Definition 2 *Let $\mathcal{P}$ be a logic program with updates. We say that a predicate p* refers to *a predicate q if there is a clause $p(\bar{t}) :- \alpha_1, ..., \alpha_i, q(\bar{s}), \alpha_{i+1}, ..., \alpha_r$ in $\mathcal{P}$. We consider the relation "*depend on*" which is the reflexive and transitive closure of the relation "refer to". Maximal strongly connected components of the graph $G(\mathcal{P})$ of the relation "depend on" are called* cliques. *A predicate (a goal, a subgoal) is* stationary *if it does not depend on elementary updates, assignments, and update predicates.*

We distinguish the following important classes of logic programs with updates.

Definition 3 *A logic program with updates $\mathcal{P}$ is* stratified *if:*

1) all query predicates are stationary;

2) let $\mathcal{P}^q$ be the set of definitions of all query predicates in $\mathcal{P}$: $\mathcal{P}^q = \mathcal{P}(\mathbf{P}^q)$. Then for any DB state $\mathcal{E}$ the logic program $\mathcal{P}^s = \mathcal{P}^q \cup \mathcal{E}$ is stratified in the sense of [1].

A logic program with updates $\mathcal{P}$ is update (u-) stratified *if it is stratified and in every clause $p(\bar{u}) :- \alpha_1, ..., \alpha_i, q(\bar{v}), \alpha_{i+1}, ..., \alpha_r$ in which p, q are update predicates belonging to the same clique of $G(\mathcal{P})$, all predicates in α_i are* stationary.

The essence of the u-stratifiability is that DB updates are available only at the steps where a clique is changed. This constraint provides upper bounds to the number of elementary DB updates fired in the course of the transaction caused by an update predicate call.

Example 1 *Let $\mathbf{P}^e = \{e/1\}$, $\mathbf{P}^q = \{d/1\}$, and $\mathbf{P}^u = \{a/0, b/0, c/0\}$. Then the following program is u-stratified:*

$a :- d(0), b.$
$a :- insert(e(0)).$
$b :- \neg d(0), a.$
$b :- d(I), I_1 := I + 1, \neg d(I_1), delete(e(I)), insert(e(I_1)), c.$
$c :- d(I), I > 0, I_1 := 2 * I, \neg d(I_1), insert(e(I_1)).$
$d(I) :- e(I).$

The dependency graph of $G(\mathbf{P}^u)$ consists of two cliques: $\{a, b\} \rightarrow \{c\}$.

Throughout this paper we consider only stratified logic programs with updates and call them simply logic programs with updates.

The operational semantics of a logic program with updates $\mathcal{P}$ is represented by a relation of the form $\mathcal{E} \cup \{\ :- u\} \overset{\theta}{\vdash}_{\mathcal{P}} \mathcal{E}' \cup \{\ :- v\}$. Intuitively it says that $\mathcal{P}$ reduces via the answer substitution θ the goal $:- u$ when in the input DB state $\mathcal{E}$, to the goal $:- v$, and changes $\mathcal{E}$ to the output DB state $\mathcal{E}'$. The following rule schemes 1 - 7 define this relation formally (index $\mathcal{P}$ is dropped). The letters u, v, ϕ in these schemes stand for sequences of subgoals, $\Box$ is for the empty goal, θ, θ', and σ are for substitutions and for an MGU, ε denotes the identity substitution, e, e' denote arithmetical expressions, A, H stand for intensional atoms, $Fact$ is for an extensional atom, and $stable\_mod(\mathcal{E}' \cup \mathcal{P}^q)$ denotes the (unique) stable model (in the sense of [7]) of the stratified program $\mathcal{E}' \cup \mathcal{P}^q$.

1. $$\frac{}{\mathcal{E} \cup \{\ :- u\} \overset{\varepsilon}{\vdash} \mathcal{E} \cup \{\ :- u\}}$$

2. $$\frac{(\mathcal{E} \cup \{\ :- u\} \overset{\theta}{\vdash} \mathcal{E}' \cup \{\ :- A, v\},\ H\ :- \phi \in \mathcal{P} \text{ and } A\circ\theta\circ\sigma = H\circ\sigma) \text{ or } (A\circ\theta\circ\sigma \in \mathcal{E}' \text{ and } \phi = \emptyset)}{\mathcal{E} \cup \{\ :- u\} \overset{\theta\circ\sigma}{\vdash} \mathcal{E}' \cup \{\ :- \phi, v\}}$$

3. $$\frac{\mathcal{E} \cup \{\ :- u\} \overset{\theta}{\vdash} \mathcal{E}' \cup \{\ :- X{:=}e, v\},\ X \text{ is free}, e\circ\theta \text{ is ground}, \theta' = \theta[X \backslash val(e,\theta)]}{\mathcal{E} \cup \{\ :- u\} \overset{\theta'}{\vdash} \mathcal{E}' \cup \{\ :- v\}}$$

4. $$\frac{\mathcal{E} \cup \{\ :- u\} \overset{\theta}{\vdash} \mathcal{E}' \cup \{\ :- e{<}e', v\},\ e\circ\theta,\ e'\circ\theta \text{ are ground and } val(e,\theta) < val(e',\theta)}{\mathcal{E} \cup \{\ :- u\} \overset{\theta}{\vdash} \mathcal{E}' \cup \{\ :- v\}}$$

5. $$\frac{\mathcal{E} \cup \{\ :- u\} \overset{\theta}{\vdash} \mathcal{E}' \cup \{\ :- insert(Fact), v\},\ Fact\circ\theta \text{ is ground}}{\mathcal{E} \cup \{\ :- u\} \overset{\theta}{\vdash} (\mathcal{E}' \cup \{Fact\circ\theta)\}) \cup \{\ :- v\}}$$

6. $$\frac{\mathcal{E} \cup \{\ :- u\} \overset{\theta}{\vdash} \mathcal{E}' \cup \{\ :- delete(Fact), v\},\ Fact\circ\theta \text{ is ground}}{\mathcal{E} \cup \{\ :- u\} \overset{\theta}{\vdash} (\mathcal{E}' \backslash \{Fact\circ\theta)\}) \cup \{\ :- v\}}$$

7. $$\frac{\mathcal{E} \cup \{\ :- u\} \overset{\theta}{\vdash} \mathcal{E}' \cup \{\ :- \neg A, v\},\ stable\_mod(\mathcal{E}' \cup \mathcal{P}^q) \models \neg A\circ\theta}{\mathcal{E} \cup \{\ :- u\} \overset{\theta}{\vdash} \mathcal{E}' \cup \{\ :- v\}}$$

These rules support leftmost strategy of subgoals evaluation. Rule 1 introduces the identity answer substitution. Rule 2 is the standard resolution step. Rules 3 and 4 deal with arithmetic. All variables in expressions e, e' should be instantiated (by numbers). Rules 3, 4 do not change DB states. $val(e, \theta)$ denotes the value of the arithmetical expression e in the environment θ. The assignment $X := e$ changes answer substitution θ binding X by $val(e, \theta)$. Rules 5 and 6 are the only rules changing DB states. They describe the effect of elementary updates. Specifically, rule 5 changes DB state $\mathcal{E}'$ to DB state $\mathcal{E}' \cup \{Fact \circ \theta)\}$. Rule 7 indicates that the negation is resolved in the (unique) stable model $stable\_mod(\mathcal{E}' \cup \mathcal{P}^q)$. It should be noted that our choice of negation semantics is quite arbitrary. Any negation semantics "effectively computable" in finite models will do here.

Rules 1 - 7 associate with each update predicate $a/0$ the following nonde-

terministic action operator $\overset{a}{\vdash_{\mathcal{P}}}$ on DB states: $\mathcal{E} \overset{a}{\vdash_{\mathcal{P}}} \mathcal{E}' \Longleftrightarrow \mathcal{E} \cup \{ \ :- \ a\} \overset{\theta}{\vdash_{\mathcal{P}}} \mathcal{E}' \cup \{\Box\}$ for some θ.
A DDB $\mathcal{B} =< \mathcal{P} \cup \{ \ :- \ a_1, \ldots, \ :- \ a_n \ \}, \ \Phi >$ includes an intensional logic program with updates $\mathcal{P}$, a predefined set of 0-ary goals $\{ \ :- \ a_1, \ldots, \ :- \ a_n\}$ implementing DB state actions, and integrity constraints (IC) embodied by a property Φ of DB states. The behavior of $\mathcal{B}$ is defined by relation $\vdash_{\mathcal{B}}$ which is the union of relations $\overset{a_i}{\vdash_{\mathcal{P}}}$, $i = 1, ..., n$. E.g., for program $\mathcal{P}$ in Example 1 $\{e(0)\} \overset{a}{\vdash_{\mathcal{P}}} \{e(0)\}$ and $\{e(0)\} \overset{a}{\vdash_{\mathcal{P}}} \{e(1), e(2)\}$ hold.

3 Problem Classes

We explore the computational complexity of the following problem. *Given a DDB* $\mathcal{B} =< \mathcal{P} \cup \{ \ :- \ a_1, \ldots, \ :- \ a_n \ \}, \ \Phi >$ *and a disturbance relation* $\overset{d}{\longrightarrow}$ *(in some representation), one should check whether* $\mathcal{B}$ *is* $\forall\exists$*-homeostatic with respect to* $\overset{d}{\longrightarrow}$ *in all DB states* $\mathcal{E}$ *which satisfy* Φ.

The three parameters of this problem - logic programs with updates, ICs and disturbance relations - are independent, and none of them is fixed. So it is readily seen that the problem is undecidable if no restrictions are imposed on these parameters. In this section we introduce various restrictions which come about naturally and guarantee the decidability of this problem.

Logic programs. We classify the restrictions to logic programs by the form of their clauses.

Definition 4 *A logic program* $\mathcal{P}$ *is* positive *if it does not use negation. It is called* ground *if all its clauses are ground. It is* flat *if all terms in its clauses are either variables or constants. We call* $\mathcal{P}$ branching *if it is not recursive, i.e. its dependency graph* $G(\mathcal{P})$ *has no cycles. And we call* $\mathcal{P}$ productional *if it defines the unique intensional predicate q/0 and all its clauses are* productions*, i.e. have the form* $q \ :- \ Con_1, \ \ldots, Con_k, Act_1, \ldots, Act_m$ *where each* Con_i *is an extensional literal and each* Act_j *is an elementary update. These are exactly the productions used in AI, so we keep their usual syntax:*

$$Con_1 \ \&...\& \ Con_m \Longrightarrow Act_1, ..., Act_n.$$

Let us denote by $Datalog^u$ the class of all *stratified* flat logic programs with updates. This class is the broadest one we consider in this paper, in which the problem of total homeostaticity is decidable. We also consider the following classes of u-stratified logic programs:

- USF is the class of u-stratified flat programs;
- USG is the class of u-stratified ground programs;
- $BRAF$ is the class of branching flat programs;
- $BRAG$ is the class of branching ground programs;
- $BRAG^p$ is the class of positive programs in $BRAG$.

Note that positive programs may use deletions.

For each class of branching programs in this list we consider the corresponding class of productional logic programs: $PROF$, $PROG$, and $PROG^p$. The last class is the smallest one. Programs in $PROG^p$ have only ground productions which do not use negation in their premises.

Integrity constraints. For a DB state $\mathcal{E}$ and an IC Φ we denote by $\mathcal{E} \models \Phi$ the relation "Φ is true on $\mathcal{E}$". The nature of IC Φ is immaterial here. What is essential is that Φ has a constructive representation in some formal language (a logical formula, a set of productions, or a polynomial time Turing machine, etc.) for which there is a universal algorithm checking that $\mathcal{E} \models \Phi$.

Definition 5 *An IC* Φ *is* preserved upwards *if whenever* $\mathcal{E} \subseteq \mathcal{E}'$ *and* $\mathcal{E} \models \Phi$, *then* $\mathcal{E}' \models \Phi$.

Below we analyze the complexity of stability problems using ICs in the following three classes.

IC_0 : Φ is preserved upwards and there is a polynomial time algorithm which enumerates all minimal DB states satisfying Φ.

IC_1 : the problem $\mathcal{E} \models \Phi$ is in **P**.

IC_2 : the problem $\mathcal{E} \models \Phi$ is in **PSPACE**.

E.g., the class IC_0 contains ICs expressed by positive ground DNFs, negation-free logical programs, and some monotone graph properties (e.g., connectivity). The well-known functional dependencies and a number of properties of model size (e.g., parity) belong to the class IC_1. The class IC_2 contains ICs expressed by the 1st order formulas, etc.

Disturbance relations. The constraints we impose on disturbance relation $\stackrel{d}{\longrightarrow}$ are expressed in terms of the following *change set:*

$c(d, \mathcal{E}) = \{(D^+, D^-) |$ *there is* $\mathcal{E}' :$ $\mathcal{E} \stackrel{d}{\longrightarrow} \mathcal{E}'$, $D^+ = \mathcal{E}' \setminus \mathcal{E}$, *and* $D^- = \mathcal{E} \setminus \mathcal{E}'\}$.

Let $\delta = (\Delta^+, \Delta^-)$ be a pair of two finite subsets of $\mathbf{B}^e$. We define a δ-*disturbance* as the relation on DB states such that for every $\mathcal{E}$ $c(\delta, \mathcal{E}) = \{(D^+, D^-) | D^+ \subseteq \Delta^+, D^- \subseteq \Delta^-\}$. Therefore, δ-disturbances specify rather primitive set theoretical bounds to the change sets, which do not depend on DB states. Whereas in all the theorems below only δ-disturbances are considered, most of them hold as well for many other classes of disturbances which reflect more detailed knowledge about the behavior of the external medium. In section 5 we describe a so called DDB factorization which extends properties of δ-disturbances to larger classes of disturbances.

Now we can formulate the general total homeostaticity problem. For a class of logic programs with updates **P**, a class of ICs **I**

$HOM^T(\mathbf{P} + \mathbf{I}) = \{(< \mathcal{P} \cup \{G\}, \Phi >, \delta) |$
$\mathcal{P} \in \mathbf{P}$, $\Phi \in \mathbf{I}, < \mathcal{P} \cup \{G\}, \Phi >$ *is* δ-$\forall\exists$-*homeostatic in all* $\mathcal{E}$ satisfying $\Phi\}$.

Example 2 *Consider the following toy example of a medical DDB.*
Let $\mathcal{P}$ *be the following* $Datalog^u$*-program:*

$cure \ :- \ disease, medicine,$
$\quad delete(disease), delete(medicine), insert(immunity).$
$instruction \ :- \ disease, \neg medicine, \neg immunity, insert(medicine).$
$sound \ :- \ \neg disease.$
$immune \ :- \ disease, immunity, delete(disease).$
Let $\Phi \equiv \neg disease \vee medicine \vee immunity$ *be the IC and* $\delta = (\{disease\}, \{immunity\})$ *be the disturbance.*

This logic program is in $PROG$, and Φ is in IC_0. It is easy to see that the DDB

$< \mathcal{P} \cup \{ \ :- \ cure, \ :- \ instruction, \ :- \ immune, \ :- \ sound\}, \Phi >$

is total homeostatic. Indeed, in any admissible DB state $\mathcal{E}$ only four δ-bounded disturbances are possible: $(\emptyset, \emptyset)$, $(\{disease\}, \emptyset)$, $(\emptyset, \{immunity\})$, $(\{disease\}, \{immunity\})$, and after any of them Φ is restored by an appropriate action. E.g.,
$\emptyset \models \Phi$ and $\emptyset \stackrel{(\{disease\},\emptyset)}{\longrightarrow} \{disease\} \not\models \Phi$. However, $\{disease\} \stackrel{instruction}{\vdash} \{disease, medicine\} \models \Phi$. If, e.g. $\{disease, medicine\} \stackrel{cure}{\vdash} \{immunity\}$, then $\{immunity\} \models \Phi$. If further $\{immunity\} \stackrel{(\{disease\},\emptyset)}{\longrightarrow} \{disease, immunity\}$, then $\{disease, immunity\} \models \Phi$, and $\{disease, immunity\} \stackrel{immune}{\vdash} \{immunity\} \models \Phi$, etc.
However, if we remove the clause "instruction" the resulting DDB won't be $\delta - \forall\exists$-homeostatic in the empty DB state $\emptyset$ which is admissible. The matter is that
there is no way to restore Φ in the DB state $\{disease\}$ without the action "instruction".

4 Complexity of total homeostaticity

In this section we establish complexity bounds to the problem of total homeostaticity in the classes of DDBs defined above. Because of space limitations most proofs are omitted here.

In the theorems to follow finite sets of facts $\mathcal{E}$, Δ^+, Δ^-, etc. are supposed to be represented in some standard coding. The size of the code $|\mathcal{E}|$ is $O(C_{\mathcal{E}} * FN_{\mathcal{E}})$, where $C_{\mathcal{E}}$ depends on maximal arity of extensional predicates and on constants used in $\mathcal{E}$, and $FN_{\mathcal{E}}$ denotes the number of facts in $\mathcal{E}$. This encoding reflects the real size of the relational DB, and differs from the encoding used in the finite model complexity where predicates are represented by the sequences of their values on *all* possible data. We use the standard notation for complexity classes. In particular, we use $SPACE(2^{poly})$ for the class of problems solvable in space $2^{p(n)}$ for some polynomial $p(n)$, $NTIME(2^{poly})$ for the class of problems solvable in nondeterministic time $2^{p(n)}$ for some polynomial $p(n)$, $TIME(2^{2^{poly}})$ for the class of problems solvable in deterministic time $2^{2^{p(n)}}$ for some polyno-

mial $p(n)$, etc.

Our first result shows that in the class of productional DDBs which don't use negation and for the ICs in the class IC_0 the problem of total homeostaticity is tractable. Note that the complex update induced by such a DDB is not monotonous in general because the productions may use deletions.

Theorem 4.1

(1) The problem $HOM^T(PROG^p + IC_0)$ *is solved in polynomial time.*
(2) The problem $HOM^T(PROG^p + IC_1)$ *is co-NP-complete.*

Sketch of the proof of (1). We enumerate all minimal DB states satisfying Φ. For each such DB state $\mathcal{E}$ we perform the maximal deletion $\mathcal{E} \setminus \Delta^- = \mathcal{E}^*$ and then fire in chain all applicable productions. The answer is "yes" if for each such DB state $\mathcal{E}^*$ there exists $\mathcal{E}_1$ such that $\mathcal{E}^* \vdash_{\mathcal{P}} \mathcal{E}_1$ and $\mathcal{E}_1 \models \Phi$. □

Theorem 4.2

(1) The problem $HOM^T(BRAG^p + IC_0)$ *is solved in polynomial time, when the size of intensional signature* $\mathbf{P}^q \cup \mathbf{P}^u$ *is bounded.*
(2) The problem $HOM^T(BRAG^p + IC_0)$ *is co-NP-complete.*

Sketch of the proof of (1). As $|\mathbf{P}^q \cup \mathbf{P}^u| \leq const$, the number of different computation trees of the DDB [3] (and therefore, the number of its actions) is bounded by a polynomial. So we can apply the algorithm used in theorem 4.1. □

The upper bounds in the theorems to follow make use of a "contraction lemma" proven in [3] (Lemma 2), which says that for each computation tree t there is an equivalent one whose size is bounded by the number of different global contexts of nodes v in t, i.e. of quadruples $\mathcal{E}(v)^{in}$ (input DB state), $\mathcal{E}(v)^{out}$ (output DB state), $\sigma(v)^{in}$ (input substitution), $\sigma(v)^{out}$ (output substitution). As a result we can evaluate the sizes of computation trees of logic programs in the classes introduced above. In particular, for flat DDBs the size of facts in DB states $FN_{\mathcal{E}}$ is bounded by $p_e c^a$ where $p_e = |\mathbf{P}^e|$, c is the number of different constants and a is the maximal arity of extensional predicates. The number of different substitution does not exceed $(c+v)^v$ where v is the maximal number of variables in clauses. Therefore, the size of a contracted tree is bounded by $2^{2^{dNlogN}}$ for some constant d and for the input length N. If the DDB is u-stratified then the height of the contracted tree, as well as the size of DB states are bounded by $2^{p(N)}$ for some polynomial $p(N)$. Therefore, the computation represented by such tree can be simulated in the space $2^{p(N)}$, which is shown in [3] (see Lemma 2). In the case where the DDB is branching the height of all its computation trees is bounded by $p_i = |\mathbf{P}^q \cup \mathbf{P}^u|$. If the extensional arity is bounded, then the size of DB states is polynomial. If moreover, the intensional signature is fixed, then the size of computation trees is bounded by a polynomial. In the ground case the size of DB states is bounded by the

input length. So if a DDB is branching or u-stratified, then its computation trees can be "simulated" in polynomial space.

Theorem 4.3

(1) If the intensional signature $\mathbf{P}^q \cup \mathbf{P}^u$ *is fixed, then the problem* $HOM^T(BRAG + IC_1)$ *is* Π_2^p*-complete.*

(2) The problem $HOM^T(BRAG + IC_1)$ *is* $PSPACE$*-complete.*

(3) If the intensional signature $\mathbf{P}^q \cup \mathbf{P}^u$ *is fixed and the extensional arity is bounded, then the problem* $HOM^T(BRAF + IC_1)$ *is* Π_2^p*-complete.*

(4) The problem $HOM^T(BRAF + IC_2)$ *is co-*$NTIME(2^{poly})$*-complete.*

Sketch of the proof. (1,3) *Lower bounds.* We reduce to our problem the well-known Π_2^p-complete problem of validity of quantified propositional formulas of the form
$\Psi = \forall x_1 \ldots \forall x_k \exists y_1 \ldots \exists y_m \phi(x_1, \ldots, x_k, y_1, \ldots, y_m)$ where $\phi(\bar{x}, \bar{y}) \in$ 3-CNF.

Given such a formula Ψ, we construct an input instance $(< \mathcal{P} \cup G, \Phi >, \delta)$ of the HOM^T-problem as follows. We set $\mathbf{P}^e = \{x_1, \ldots, x_k, y_1, \ldots, y_m\}$, $\mathbf{P}^u = \{g, f\}$. The program $\mathcal{P}$ has the clause
$g \text{ :- } f, f, \ldots, f \quad (m \text{ occurrences})$,
and for every $1 \leq j \leq m$ it includes two clauses:
$f \text{ :- } insert(y_j)$ and $f \text{ :- } delete(y_j)$.
Let $\phi(x_1, \ldots, x_k, y_1, \ldots, y_m)$ serve as the IC, and $\delta = (\emptyset, \emptyset)$. Then Ψ is valid iff $(< \mathcal{P} \cup \{ \text{ :- } g\}, \Phi >, \delta) \in HOM^T(BRAG + IC_1)$.

(2)*Lower bound.* For a nondeterministic Turing machine $\mathcal{M}$ with m tape symbols and r states, whose space is bounded by some polynomial $p(n)$, n being the length of an input word, we set $l = [log_2 T] + 1$, where $T = N(r+1)(m+1)^N$, $N = p(n)$. The program $\mathcal{P}$ has the following clauses.
$g \text{ :- "}accept\text{"}$
$g \text{ :- } p_1$
$p_i \text{ :- } p_{i+1}, p_{i+1} \quad (1 \leq i \leq l-1)$
$p_l \text{ :- } s, s.$
Besides them for each instruction $q_j a_i \rightarrow q_u a_v S \quad (S \in \{-1, 0, 1\})$ it includes the set of clauses
$s \text{ :- } q_j, h_k, a_{k,i}, delete(q_j), delete(h_k), delete(a_{k,i}), insert(q_u), insert(a_{k,v}),$
$\quad insert(h_{k+S})$ for each $1 \leq k \leq N$.
The unique goal is $\text{ :- } g$, and $\delta = (\emptyset, \emptyset)$. Finally, the IC is $\Phi =$ "$success$" $\vee INIT$ where "success" expresses that a DB state represents a successful final instantaneous description, and $INIT$ expresses that a DB state represents the initial instantaneous description with the input x. The theorem follows from the fact that $\mathcal{M}$ accepts x iff $(< \mathcal{P} \cup \{ \text{ :- } g\}, \Phi >, \delta) \in HOM^T(BRAG + IC_1)$.

(4)*Lower bound.* We show that the complement of $HOM^T(PROF + IC_0)$ is $NTIME(2^{poly})$-hard. We fix a nondeterministic Turing machine $\mathcal{M}$ with m tape symbols and r states which works in time bounded by $2^{p(n)}$ for some polynomial $p(n)$, and we set $N = p(n) + 1$. Let

$a(t_1, \ldots, t_N, s_1, \ldots, s_N, i)$ mean that the cell with the binary number $s_1 \ldots s_N$ ($s_i \in \{0, 1\}$) at the moment of time with the binary number $t_1 \ldots t_N$ contains the symbol a_i. Let $h(t_1 \ldots t_N, s_1, \ldots, s_N, k)$ mean that at the moment $t_1 \ldots t_N$ the head of $\mathcal{M}$ visits the cell $s_1 \ldots s_N$ in the state q_k .

Let $\Phi =$ "*reject*" and $\delta = (\emptyset, \emptyset\})$. Then the program $\mathcal{P}$ consists of the following six groups of productions.

G1: Productions checking that there is a moment when some cell is empty or contains two different symbols. We present them as an example:

$\neg a(\bar{T}, \bar{S}, 0)\ \& \ldots \&\ \neg a(\bar{T}, \bar{S}, m) \Longrightarrow insert(reject)$

$a(\bar{T}, \bar{S}, i)\ \&\ a(\bar{T}, \bar{S}, j) \Longrightarrow insert(reject)$ (for all $0 \leq i < j \leq m$).

G2: Productions checking that there is a moment when the uniqueness of the head position or of the state is violated.

G3: Productions checking that the facts of a DB state do not represent the initial instantaneous description of $\mathcal{M}$ with the input x at the moment of start.

G4: Productions checking that there is a moment $\bar{t}$ when a symbol is changed in a cell not visited by the head.

G5: Productions checking that there is a moment $\bar{t}$ when the position or the state of the head, or the symbol against the head are changed incorrectly (i.e. the facts for some moments $\bar{t}$ and $t \dot{+} 1$ do not fit any instruction of $\mathcal{M}$).

The productions in groups G1 - G5 check that a DB state does not represent an accepting computation of $\mathcal{M}$ with input x, and if this is the case insert the fact $reject$ into the DB state.

The last two productions check that a computation represented by a DB state is successful.

G6: $h(1, 0, \ldots, 0, 0, \ldots, 1, "no") \Longrightarrow insert(reject)$.

$h(1, 0, \ldots, 0, 0, \ldots, 1, "accept") \Longrightarrow delete(reject)$. $(*)$

Thus, the production (*) is the only one which leads to DB states violating Φ. The lower bound is implied by the following assertion.

Lemma 4.4 *$\mathcal{M}$ accepts x* $\Longleftrightarrow$ $(< \mathcal{P}, \Phi >, \delta) \notin HOM^T(PROF + IC_0)$.

□

The proof of the point (2) of Theorem 4.3 shows as well that the problem $HOM^T(USG + IC_2)$ is $PSPACE$-hard. So we can state now that this problem is $PSPACE$-complete. As to the problems $HOM^T(USF + IC_2)$, $HOM^T(Datalog^u + IC_2)$ and $HOM^T(Datalog^u + IC_2)$ the proofs of their lower bounds are rather technical and lengthy and are omitted.

Theorem 4.5

(1) The problem $HOM^T(USG + IC_2)$ is $PSPACE$-complete.

(2) The problem $HOM^T(USF + IC_2)$ is $SPACE(2^{poly})$-complete.

Theorem 4.6

The problem $HOM^T(Datalog^u + IC_2)$ is $TIME(2^{2^{poly}})$-complete.

5 DB factorizations

In real application databases data can as a rule be naturally stratified into classes where specific values of certain attributes are purely informational, i.e. their change within these classes does not affect neither integrity constraints, nor main operational properties. The data differing only in such "insignificant" features can be regarded as equivalent. E.g., if a company database represents contracts by extensional facts of the form $contract(Number, Date, Customer, Sum)$, then it is natural to consider equivalent all facts $contract(i, d, c, s)$ differing only in customer names c, because these names are not relevant from the financial point of view. To be more precise, such equivalences are not absolute. They are induced by specific data retrieval or analysis procedures. E.g., if there is a need to find out a cause of queues in a library on the ground of dynamic records of rendered services, of the form:

$report(Y, M, D, Hour, Min, Service, ClientId, ShelfMark, InCharge)$
it is natural to suppose two records $report(R_1), report(R_2)$ equivalent if their projections onto the attributes $Hour, Min, Service, InCharge$ coincide. In the situations like ours where one should verify certain global properties of data defined through IC it is natural to factorize the data with respect to an equivalence compatible with the IC. Such data factorization sometimes permits reduction of a large and potentially infinite application domain to a bounded and transparent one.

Definition 6 *Let* $\equiv$ *be an equivalence on* $\mathbf{H}$. *It can be extended naturally onto* $\mathbf{B}^e$: $g(t_1, ..., t_n) \equiv g(t'_1, ..., t'_n)$ *iff* $t_i \equiv t'_i$ *for all* $1 \leq i \leq n$. *For every* $D_1, D_2 \subseteq \mathbf{B}^e$ *we set* $D_1 \Leftarrow\!\!\!| D_2$ *if* $D_1/_\equiv \subseteq D_2/_\equiv$. $D_1 \Leftrightarrow D_2$ *if* $D_1 \Leftarrow\!\!\!| D_2$ *and* $D_2 \Leftarrow\!\!\!| D_1$.

A binary relation $\Re$ *is* compatible with an equivalence $\equiv$ *if for any three DB states* $\mathcal{E}_1, \mathcal{E}'_1, \mathcal{E}_2 \subseteq \mathbf{B}^e$ *such that* $\mathcal{E}_1 \Leftrightarrow \mathcal{E}_2$ *and* $\mathcal{E}_1 \Re\, \mathcal{E}'_1$, *there exists such a DB state* $\mathcal{E}'_2$ *that* $\mathcal{E}'_1 \Leftrightarrow \mathcal{E}'_2$ *and* $\mathcal{E}_2 \Re \mathcal{E}'_2$.

A formula Φ *is* compatible with an equivalence $\equiv$ *if for any DB states* $\mathcal{E}_1, \mathcal{E}_2 \subseteq \mathbf{B}^e$ $\mathcal{E}_1 \Leftrightarrow \mathcal{E}_2$ *implies* $\mathcal{E}_1 \models \Phi$ *iff* $\mathcal{E}_2 \models \Phi$.

We can therefore factorize DDBs as well.

Definition 7 *A (*factorized*) DDB is a triple* $\mathcal{B} =< \mathcal{P} \cup \{ :- a_1, \ldots, :- a_n\}, \Phi, \equiv >$, *where* $\equiv$ *is some polynomial time computed equivalence relation on* $\mathbf{H}$ *and all logic program action relations* $\vdash^{a_i}_{\mathcal{P}}$ *and the IC* Φ *are compatible with* $\equiv$.

Let $\delta = (\Delta^+, \Delta^-)$ *be a pair of two finite subsets of* $\mathbf{B}^e$. *We can weaken the definition of* δ*-disturbance as follows.* $\delta_\equiv$*-disturbance is the relation on DB states such that for every* $\mathcal{E}$ $c(\delta_\equiv, \mathcal{E}) = \{(D^+, D^-) | D^+ \Leftarrow\!\!\!| \Delta^+, D^- < \equiv \Delta^-\}$. *We call two trajectories of* $\mathcal{B}$ equivalent *if their DB states with the same ordinal numbers are equivalent.*

The following technical lemma reduces in many cases the space of all $\delta_{\equiv}$-bounded trajectories to a subspace of δ-bounded trajectories.

Lemma 5.1 *Let $\mathcal{B}$ be a DDB with ground (flat) intensional logic program, $\mathcal{E}$ be some its DB state and $\delta =< \mathcal{D}^+, \mathcal{D}^- >$ with $\mathcal{D}^+, \mathcal{D}^- \subseteq \mathbf{B}^e$ being finite. Then there is a finite $\tilde{\delta} =< \tilde{\mathcal{D}}^+, \tilde{\mathcal{D}}^- >$ such that for every $\delta_{\equiv}$-trajectory ω of $\mathcal{B}$ starting in $\mathcal{E}$ there is an equivalent $\tilde{\delta}$-trajectory $\tilde{\omega}$ of $\mathcal{B}$ starting in $\mathcal{E}$. In addition, if $\mathcal{B}$ is ground, then $\mid \tilde{\delta} \mid = O(\mid \mathcal{B} \mid + \mid \mathcal{E} \mid + \mid \delta \mid)$, and if $\mathcal{B}$ is flat, then $\mid \tilde{\delta} \mid \leq 2^{pol(|\mathcal{B}|+|\mathcal{E}|+|\delta|)}$.*

Lemma 5.1 can be used for the purpose of extending all the proofs of upper bounds in the theorems we have proven to the corresponding theorems for factorized DDBs with disturbances $\delta_{\equiv}$. In these proofs we find algorithms constructing or guessing a homeostatic $\delta_{\equiv}$-trajectory. Application of this lemma guarantees that such a trajectory exists iff there is a $\tilde{\delta}$-trajectory with the same property. The given input pair $< \mathcal{B}, \delta >$ is therewith constructively changed to $< \mathcal{B}, \tilde{\delta} >$. However, as shows the proof of lemma 5.1, $\tilde{\delta}$ is constructed in polynomial time in the ground case and in exponential time in the flat case. Therefore, due to this lemma without loss of generality we can describe algorithms treating only homeostatic δ-trajectories. Note that $c(\delta, \mathcal{E})$ is always finite, whereas $c(\delta_{\equiv}, \mathcal{E})$ can be infinite.

Conclusion

The proposed concept of the total homeostatic behavior of DDBs substantially generalizes the following property: "for any given external update violating the IC there exists a reaction of the data base, which restores the IC". This concept applies to the analysis of the behavior of discrete dynamic systems with complex states, functioning in active medium, such as complex imbedded hardware, or biological systems, or interactive systems whose operation involves consumption, compensation, and restoration of some resources, e.g. plants, trade enterprises, warehouses, etc. In this paper we consider only the homeostatic behavior, because of its close relation to data base scenarios. Meanwhile, in other application domains we encounter other types of steady behavior e.g. stable or perspective one, which have been formalized and explored in our previous publications [3, 4, 5]. We consider rather broad classes of DDBs. So not surprisingly the complexity of the property of homeostaticity in these classes is sometimes very high. However, in real applications the intensional specification of system behavior is composed from a variety of independent small modules. For such DDBs our results guarantee quite tractable upper complexity bounds. Very efficient CAD systems can be created on the ground of the proposed concepts, which can support an interactive experimental analysis of behavior of complex discrete dynamic systems in broad classes of applications.

References

[1] K. R. Apt, H. Blair, and A. Walker. Towards a theory of declarative knowledge. In: J. Minker (ed.) *Foundations of deductive databases and logic programming. Morgan Kaufman Pub., Los Altos*, 89–148, 1988.

[2] U. Dayal, E. Hanson, and J. Widom. Active database systems. In: W. Kim (ed.) *Modern Database Systems, Addison Wesley*, 436–456, 1995.

[3] M. I. Dekhtyar, A. Ja. Dikovsky. Dynamic Deductive Data Bases with Steady Behavior. In: L. Sterling (ed.) *Proc. of the 12th International Conf. on Logic Programming. The MIT Press*, 183–197, 1995.

[4] M. I. Dekhtyar, A. Ja. Dikovsky. On Homeostatic Behavior of Dynamic Deductive Data Bases. In: D. Bjorner, M. Broy, and I. Pottosin (eds.) *Proc. 2nd Int. A.P.Ershov Memorial Conf. Perspectives of System Informatics. LNCS*, 1181:420–432, 1996.

[5] M. I. Dekhtyar, A. Ja. Dikovsky. Properties of Steady Behavior of Dynamic Deductive Data Bases. Part II. Homeostaticity. *Techn. rept. Universite Paris XII-Val de Marne*, 96-09:1–15, Juin 1996.

[6] T. Eiter, G. Gottlob. On the complexity of propositional knowledge base revision, updates, and counterfactuals. *Artificial Intelligence*, 57:227–270, 1992.

[7] M. Gelfond, V. Lifschitz. The stable semantics for logic programs. In: R. Kovalsky and K. Bowen (eds) *Proc. of the 5th Intern. Symp. on Logic Programming. Cambridge, MA, MIT Press*, 1070–1080, 1988.

[8] G. Gottlob, G. Moercotte, V. S. Subrahmanian. The PARK semantics for Active Databases. In: *Proc. of EDBT'96.* Avignon, France, 1996.

[9] M. Halfeld Ferrari Alves, D. Laurent, N. Spyratos. Update rules in Datalog programs. *Rapport de Recherche Université de Paris-Sud, Centre d'Orsay, LRI*, 1024, Janvier 1996.

[10] N. Katsuno, A.O. Mendelzon. Propositional knowledge base revision and minimal change. *Artificial Intelligence*, 52:253–294, 1991.

[11] V. W. Marek, M. Truszciński. Revision programming, database updates and integrity constraints. In: *International Conference on Data Base theory, ICDT, LNCS*, 893:368-382, 1995.

[12] T. C. Przymusinski, H. Turner. Update by Means of Inference Rules. In: V.W. Marek, A. Nerode, M. Truszczyński (eds) *Logic Programming and Nonmonotonic Reasoning, Proc. of the Third Int. Conf. LPNMR'95, Lexington, KY, USA*, 166–174, 1995.