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 their 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 specific DB state. In particular, 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. a finite set of extensional facts. Possible actions (transitions) of the system in given states can be represented by some binary relation $\vdash$. External stimuli of the active medium of the system (*disturbances*) also change states and can be represented by another binary relation $\stackrel{d}{\longrightarrow}$. In our approach the fundamental difference between these two relations is determined by the level of our knowledge about them. The theory of behavior of the system is assumed to be known. This being so, we describe it in the form of a logic program with updates $\mathcal{P}$ over DB states. More specifically, a transition from state $\mathcal{E}_1$ to state $\mathcal{E}_2$, i.e. the action $\mathcal{E}_1 \vdash \mathcal{E}_2$ is described as a computation of some predefined goal $:- a$ of $\mathcal{P}$ transforming the first DB state into the second: $\mathcal{E}_1 \stackrel{a}{\vdash_{\mathcal{P}}} \mathcal{E}_2$. As for the disturbances, one can always estimate the maximal possible disturbance, and evaluate the effect of a committed disturbance on the current state. However, *one cannot predict in the given state which disturbance is to be committed.* E.g., in a bank there exists the list of feasible services required by its clients (disturbances), and the corresponding transactions (actions), but one cannot predict exactly which orders for the services emerge at the given moment.

Local behavior of the system in the 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

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

comes first, a medium disturbance or a system action: $\mathcal{E}_0 \xrightarrow{d} \mathcal{E}_1^* \overset{a}{\vdash}_{\mathcal{P}} \mathcal{E}_1$ and $\mathcal{E}_0 \overset{a}{\vdash}_{\mathcal{P}} \mathcal{E}_1^* \xrightarrow{d} \mathcal{E}_1$. 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. We represent the admissibility criterion by some *integrity constraints (IC)* 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 $\mathcal{E}_0 \xrightarrow{d} \mathcal{E}_1^* \overset{a}{\vdash}_{\mathcal{P}} \mathcal{E}_1$ or $\mathcal{E}_0 \overset{a}{\vdash}_{\mathcal{P}} \mathcal{E}_1^* \xrightarrow{d} \mathcal{E}_1$ is *feasible* if $\mathcal{E}_0 \models \Phi$ and $\mathcal{E}_1 \models \Phi$. Thus, the discrete dynamic system is represented by the *deductive data base (DDB)* $\mathcal{B} =< \mathcal{P} \cup GOALS,\ \Phi >$, its medium is described by relation $\xrightarrow{d}$, and their interaction is expressed in terms of feasible interactions.

Global behavior of the system in current 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

$$\mathcal{E}_0 \xrightarrow{d} \mathcal{E}_1^* \overset{a_1}{\vdash} \mathcal{E}_1 \xrightarrow{d} \mathcal{E}_2^* \overset{a_2}{\vdash} \mathcal{E}_2 \ ...$$

and *action-disturbance* trajectories

$$\mathcal{E}_0 \overset{a_1}{\vdash} \mathcal{E}_1^* \xrightarrow{d} \mathcal{E}_1 \overset{a_2}{\vdash} \mathcal{E}_2^* \xrightarrow{d} \mathcal{E}_2 \ ...$$

Infinite feasible disturbance-action trajectories represent the homeostatic behavior of the DDB in spite of disturbances of its medium: the DDB always manages to restore the IC along this trajectory if and when they are violated by the medium disturbances. Such trajectories are called *homeostatic*. Dually, the infinite feasible action-disturbance trajectories represent the stable behavior of the DDB owing to disturbances of its medium: the medium always compensates along the trajectory for possible ruinous actions of the DDB. They are called *stable*.

Trajectories of each type form a tree 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 $\xrightarrow{d}$ 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 [4] we introduce $\exists\exists$-stability and explore its computational complexity. In [5] 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 [6].

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

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

It is equivalent to $\forall\exists$-homeostaticity in any admissible DB state. This property has attracted 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. [11]) are applied. In fact the same transformation is investigated in [12, 13] in more general context. Here 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 [8]), "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 [10] 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 enforcing the IC correspond to restoring DDB actions in our terms. This is in fact the basic scenario in active data bases (cf. [3, 9]), where the disturbances are either the 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 certain class of DDBs* $\mathbf{P}$, *an IC* Φ *in a class of ICs* $\mathbf{I}$, *and a disturbance relation* $\xrightarrow{d}$ *restricted in some sense, one should* **recognize** *whether* $\mathcal{B}$ *is totally homeostatic with respect to* $\xrightarrow{d}$.

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 many 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 otherwise 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 specific DB

state. E.g., as we have shown in [5], already in the class of ground positive production systems the homeostaticity in a specific DB state with respect to polynomial time ICs is $TIME(2^{poly})$-complete.

Although in some interesting classes 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 $\mathbf{B}^e$, $\mathbf{B}^i$ the extensional and intensional Herbrand bases. *DB states* are finite subsets of $\mathbf{B}^e$. We consider *logic programs with updates* in $\mathbf{L}$ with clauses of the form $Head :- Body$, 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 $\mathbf{F}$. 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* $\mathbf{P}^e = \{e/1\}$ *and* $\mathbf{P}^u = \{a/0, b/0, c/0\}$. *Then the following program is u-stratified:*
$a \ :\!-\ e(0), b.$
$a \ :\!-\ insert(e(0)).$
$b \ :\!-\ \neg e(0), a.$
$b \ :\!-\ e(I), I_1 := I + 1, \neg e(I_1), delete(e(I)), insert(e(I_1)), c.$
$c \ :\!-\ e(I), I > 0, I_1 := 2 * I, \neg e(I_1), insert(e(I_1)).$

Here the dependency subgraph on $G(\mathbf{P}^u)$ 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 [8]) 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 nondeterministic 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.

It is readily seen that this problem is undecidable if no restrictions are imposed on its main parameters: logic programs with updates, ICs and disturbance relations. 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.

Every class of branching programs has its productional counterpart: $PROF$, $PROG$, 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., 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 class IC_1. Class IC_2 contains ICs expressed by the 1st order formulas, etc.

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

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

Let $\delta = (\Delta^+, \Delta^-)$ be a pair of two finite subsets of $\mathbf{B}^e$. We define the 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. E.g., the following k-δ-disturbances reflect the knowledge about the size of change sets: $c(k\text{–}\delta, \mathcal{E}) = \{(D^+, D^-) | D^+ \subseteq \Delta^+, D^- \subseteq \Delta^-, \ |D^+| \leq k, |D^-| \leq k\}$. Another broad class of specific δ-disturbances are the A-δ-disturbances recognized by polynomial time algorithms A : $c(A\text{–}\delta, \mathcal{E}) = \{(D^+, D^-) | D^+ \subseteq \Delta^+, D^- \subseteq \Delta^-, \ A(\mathcal{E}, D^+, D^-) = TRUE\}$. These classes reflect some features of external medium effect in the case where the medium has an open access to DB states. In section 5 we describe so called DDB factorization which extends properties of δ-disturbances to larger classes of disturbances.

We finish this section by a definition of total homeostaticity problem. For a glass of logic programs with updates **P**, a class of ICs **I**

$HOM^T(\mathbf{P} + \mathbf{I}) = \{(< \mathcal{P} \cup \{G\}, \Phi >, \delta) \mid$

$\mathcal{P} \in \mathbf{P}, \ \Phi \in \mathbf{I}, < \mathcal{P} \cup \{G\}, \Phi > \textit{ is } \delta\text{-}\forall\exists\textit{-homeostatic in all } \mathcal{E}$ satisfying $\Phi\}$.

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

$cure \ :- \ disease, medicine, delete(disease), delete(medicine), insert(immunity).$
$instruction \ :- \ disease, \neg medicine, \neg immunity, insert(medicine).$
$sound \ :- \ \neg disease.$
$immune \ :- \ disease, immunity, delete(desease).$

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 $\emptyset \overset{(\{disease\},\emptyset)}{\longrightarrow} \{disease\} \not\models \Phi$. However, $\{disease\} \overset{instruction}{\vdash} \{disease, medicine\} \models \Phi$. If, e.g. $\{disease, medicine\} \overset{cure}{\vdash} \{immunity\}$, then $\{immunity\} \models \Phi$. If further $\{immunity\} \overset{(\{disease\},\emptyset)}{\longrightarrow} \{disease, immunity\}$, then $\{disease, immunity\} \models \Phi$, and $\{disease, immunity\} \overset{immune}{\vdash} \{immunity\} \models \Phi$, 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 some proofs are omitted here, the others are only sketched.

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 is different from one used in finite model complexity where predicates are represented by the sequences of their values on *all* possible data. We use 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 polynomial $p(n)$, 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 traatable. Note that the update induced by such DDB is not monotonous in general because the productions may use deletions.

Theorem 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 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 [4] (and therefore, the number of its actions) is bounded by a polynomial. So we can apply the algorithm used in theorem 1. □

The upper bounds in the theorems to follow make use of the "contraction lemma" proven in [4] (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 unifier), $\sigma(v)^{out}$ (output unifier). As a result we can evaluate the size of computation trees in the introduced classes of logic programs. 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 unifiers does not exceed $(c+v)^v$ where v is the maximal number of variables in clauses. Therefore, the size of contracted tree is bounded by $2^{2^{dNlogN}}$ for some constant d and 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 [4] (see Lemma 2). In the case where the DDB is branching the height of computation trees is bounded by $p_i = |\mathbf{P}^q \cup \mathbf{P}^u|$. When the extensional arity is bounded the size of DB states is polynomial. If moreover the intensional signature is fixed the size of the computation tree is polynomial. In ground case the size of DB states is bounded by the input length. So if the DDB is branching or u-stratified its computation trees can be "simulated" in polynomial space. After this remark only lower bounds are sketched in the theorems to follow.

Theorem 3

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

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

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

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

Sketch of the proof. (1,3) *Lower bound.* 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 :– $f, f, \ldots, f$ (m occurrences),
and for every $1 \leq j \leq m$ it includes two clauses:
f :– $insert(y_j)$
f :– $delete(y_j)$.
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.* Let us fix a nondeterministic Turing machine $\mathcal{M}$ which works in space bounded by some polynomial $p(n)$, n being the length of an input word $x = a_{i_1} a_{i_2} ... a_{i_n}$. We set $N = p(n)$. Let $A = \{a_0, a_1, \ldots, a_m\}$ be the tape alphabet and $Q = \{q_0, \ldots, q_r\}$ be the state alphabet of $\mathcal{M}$ where q_0 is the initial state, $q_{r-1} =$ "no" is the rejecting state, and $q_r =$ "yes" is the accepting state. We assume that at the last step the head of $\mathcal{M}$ visits the first cell in one of its final states "yes" or "no". We choose the extensional signature $\mathcal{P}^e = Q \cup \{h_j \mid 1 \leq j \leq N\} \cup \{a_{j,k} \mid 1 \leq j \leq N, 0 \leq k \leq m\}$. These predicates describe the state and the position of the head, and the symbols written in the tape cells. The computation time of $\mathcal{M}$ with the input x is bounded by $T = N(r+1)(m+1)^N$. We set $l = [log_2 T] + 1$ and choose $\{g, p_1, \ldots, p_l, s\}$ as the intensional signature. The program $\mathcal{P}$ has the following clauses.
g :– "yes"
g :– p_1
p_i :– p_{i+1}, p_{i+1} $(1 \leq i \leq l-1)$
p_l :– s, s.
Besides them for each instruction $q_j a_i \rightarrow q_u a_v S$ $(S \in \{-1, 0, 1\})$ it includes the set of clauses
s :– $q_j, h_k, a_{k,i}, delete(q_j), delete(h_k), delete(a_{k,i}), insert(q_u), insert(a_{kv}), insert(h_{k+S})$
for $1 \leq k \leq N$.
The unique goal is :– g. $\delta = (\emptyset, \emptyset)$. Finally, the IC is $\Phi =$ "yes" $\vee\, INIT$ where "yes" says that a DB state represents a successful final instantaneous description, and $INIT$ says that a DB state represents the initial instantaneous description with 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.* In fact, we show that the complement of $HOM^T(PROF + IC_0)$ is $NTIME(2^{poly})$-hard. Let us fix a nondeterministic Turing machine $\mathcal{M}$ which works in time bounded by $2^{p(n)}$ for some polynomial $p(n)$, and keep the assumptions and notation of the preceding point. Let $N = p(n) + 1$, and $\mathcal{P}^e = \{bad/0, a/2N+1, h/2N+1\}$. bad indicates that a DB state does not represent an accepting computation of $\mathcal{M}$ with input x. $a(t_1, \ldots, t_N, s_1, \ldots, s_N, i)$ means that the cell with binary number $s_1 \ldots s_N$ ($s_i \in \{0, 1\}$) at the moment of time with binary number $t_1 \ldots t_N$ contains the symbol a_i. $h(t_1 \ldots t_N, s_1, \ldots, s_N, k)$ means 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 $\Phi = bad$ and $\delta = (\emptyset, \emptyset\})$. The program $\mathcal{P}$ is formed of six groups of productions which we just sketch. 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 bad into the DB state.
G1: Productions checking that there is a moment when some cell is empty or contains two different symbols. We present them as an example:
$\neg a(\bar{T}, \bar{S}, 0)\ \&\ldots\&\ \neg a(\bar{T}, \bar{S}, m) \Longrightarrow insert(bad)$
$a(\bar{T}, \bar{S}, i)\ \&\ a(\bar{T}, \bar{S}, j) \Longrightarrow insert(bad)$ (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 at the initial moment facts of DB state do not represent the initial instantaneous description of $\mathcal{M}$ with input x.
G4: Productions checking that there is a moment 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 (the facts at the next moment do not correspond to any instruction of $\mathcal{M}$).

The last two productions check that a computation represented by a DB state is successful.
G6: $h(1, 0, \ldots, 0, 0, \ldots, 1, "no") \Longrightarrow insert(bad)$.
$h(1, 0, \ldots, 0, 0, \ldots, 1, "yes") \Longrightarrow delete(bad)$. $(*)$
Thus, the production (*) is the only one which leads to DB states violating Φ. The lower bound is implied by the following assertion.

Lemma 1 *$\mathcal{M}$ accepts x $\Longleftrightarrow$ $(< \mathcal{P}, \Phi >, \delta) \notin HOM^T(PROF + IC_0)$.*

($\Rightarrow$) Let DB state $\mathcal{E}$ include all the facts representing a successful computation of $\mathcal{M}$ with input x and the fact bad. Then $\mathcal{E} \models \Phi$. After the empty update the only applicable production is (*), and it deletes bad from the state. Therefore, the result $\mathcal{E} \setminus \{bad\} \not\models \Phi$ and $\{(< \mathcal{P}, \Phi >, \delta)$ is not $\forall\exists$-homeostatic in $\mathcal{E}$.
($\Leftarrow$) Let us assume that there is a DB state $\mathcal{E}$ such that $\mathcal{E} \models \Phi$ and for any $\mathcal{E}^*$ if $\mathcal{E} \vdash_{\mathcal{P}} \mathcal{E}^*$ implies $\mathcal{E}^* \not\models \Phi$. Then $bad \in (\mathcal{E} \setminus \mathcal{E}^*)$ and therefore, only production (*) is applicable to $\mathcal{E}$. Then the productions in G3 ensure that $\mathcal{E}$ includes all the facts representing the initial instantaneous description of $\mathcal{M}$ with input x. The productions in G4 and in G5 ensure (by induction) that $\mathcal{E}$ includes all the facts representing some computation of $\mathcal{M}$ with input x, which has no more than 2^{N-1} steps, and productions in G1 and in G2 guarantee that $\mathcal{E}$ contains no extra facts. Since the production (*) is applicable to $\mathcal{E}$, $h(1, 0, \ldots, 0, 0, \ldots, 1, "yes") \in \mathcal{E}$, and therefore, the computation of $\mathcal{M}$ with input x, represented by $\mathcal{E}$ is successful. □

In the proof of point (2) of Theorem 3 we prove 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 $HOM^T(USF + IC_2)$ and $HOM^T(Datalog^u + IC_2)$ the proofs of their lower bounds are rather tedious and are omitted here.

Theorem 4
(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 5
The problem $HOM^T(Datalog^u + IC_2)$ *is* $TIME(2^{2^{poly}})$*-complete.*

Proof. Lower bound. At first we show how $DATALOG^u$-programs can compute arithmetical functions over arguments of an exponential size using predicates of polynomial arity and the bounded set of constants.

Definition 6 *Let* $M = 2^m$ *and* $a_{M-1}\dots a_1a_0$, $a_i \in \{0,1\}$, *be a binary representation of an integer* a, $0 \le a < 2^{2^m}$, *and* $p^{(m)} \in \mathbf{P}^e$. *A DB state* $\mathcal{E}$ represents a *by* p *iff for every* j, $0 \le j \le 2^m - 1$, $a_j = 1 \iff p(j_0, j_1, ..., j_{2^m-1}) \in \mathcal{E}$ *for binary representation* $j_0, j_1, ..., j_{2^m-1}$ *of* j.

E.g. the state $\mathcal{E} = \{p(0,0), p(0,1), p(1,1)\}$ represents 11 by $p^{(2)}$, and empty state $\mathcal{E}_0 = \emptyset$ represents 0.

Definition 7 *Let* $f(x_1, ..., x_n)$ *be an arithmetic function,* $p_1, ..., p_m, r$ *are an extensional predicates (* r *may be one of* $p_1, ..., p_n$ *). A* $DATALOG^u$*-program* $\mathcal{P} \cup \{ :- g\}$ computes f from $p_1, ..., p_n$ to r on interval $[0, N]$ *if for any* n*-tuple* $(a_1, ..., a_n)$ *such that* $a_j \in [0, N]$ *(* $1 \le j \le n$ *) and* $f(a_1, ..., a_n) \in [0, N]$, *and for every DB state* $\mathcal{E}$ *which represents* a_j *by* p_j $(1 \le j \le n)$ *update* g *determinisiticaly transforms* $\mathcal{E}$ *into such DB state* $\mathcal{E}_1$ *that represents* $f(a_1, ..., a_n)$ *by* r.

The following program $\mathcal{P}_1 \cup \{ :- add1\}$ computes function $s(x) = x + 1$ from p to p on interval $[0, 2^{2^m} - 2]$.
$\mathcal{P}_1$:
$add1 \ :- \ g(0, 0, ..., 0).$
$g(X_1, ..., X_m) \ :- \ \neg p(X_1, ..., X_m), \ insert(p(X_1, ..., X_m)).$
$g(X_1, ..., X_m) \ :- \ p(X_1, ..., X_m), \ delete(p(X_1, ..., X_m),$
$next(X_1, ..., X_m, Y_1, ..., Y_m), \ g(Y_1, ..., Y_m).$
The intensional predicate $next^{(2m)}(X_1, ..., X_m, Y_1, ..., Y_m)$ *given input values* $X_1, ..., X_m$ *representing the binary number* x *returns output values* $Y_1, ..., Y_m$ *representing the number* $y = x + 1$ *for every* $x < 2^m - 1$.
$next(X_1, ..., X_{m-1}, 0, X_1, ...X_{m-1}, 1).$
$next(X_1, ..X_{m-2}, 0, 1, X_1, ...X_{m-2}, 1, 0).$
$next(X_1, ..X_j, 0, 1, ..1, X_1, ...X_j, 1, 0, ..., 0).$
$next(1, ...1, 0, ..., 0).$
The following program $\mathcal{P}_2 \cup \{ :- eq\_pr\}$ checks the equality of two numbers representing by predicates $p^{(m)}$ and $q^{(m)}$ on interval $[0, 2^{2^m} - 1]$. The result is represented by 0-ary predicate $yes^{(0)}$.
$\mathcal{P}_2$:
$eq\_pq \ :- \ eq\_pq(0, ..., 0).$
$eq\_pq(X_1, ..., X_{j-1}, 0, X_{j+1}, ..., X_m) \ :- \ p(X_1, ..., X_m), q(X_1, ..., X_m),$

$next(X_1,...,X_m,Y_1,...,Y_m), eq\_pq(Y_1,...,Y_m)$ $(1 \le j \le m)$
$eq\_pq(X_1,...,X_{j-1},0,X_{j+1},...,X_m)$:- $\neg p(X_1,...,X_m), \neg q(X_1,...,X_m),$
$next(X_1,...,X_m,Y_1,...,Y_m), eq\_pq(Y_1,...,Y_m)$ $(1 \le j \le m)$
$eq\_pq(X_1,...,X_m)$:- $p(X_1,...,X_m), \neg q(X_1,...,X_m), delete(yes).$
$eq\_pq(X_1,...,X_m)$:- $\neg p(X_1,...,X_m), q(X_1,...,X_m), delete(yes).$
$eq\_pq(1,...,1)$:- $p(1,...,1),\ q(1,...,1),\ insert(yes)$
$eq\_pq(1,...,1)$:- $\neg p(1,...,1),\ \neg q(1,...,1),\ insert(yes)$
Similarly one can construct programs for arithmetical operations and relations $+, -, \times, :$, $<$, etc. The following assertion shows that the class of arithmetical functions computable by $DATALOG^u$-programs on DB states is rich enough.

Lemma 2 *Let $f(x_1,...,x_k)$ be an arithmetic function computable by some Turing machine $\mathcal{M}$ in linear space, i.e. $s_{\mathcal{M}}(x_1,...,x_k) \le c(|x_1| + ... + |x_k|)$ for some constant c and all inputs $x_1,...,x_k$. Then for every n there exist a $DATALOG^u$-program $\mathcal{P}_n$ such that*
(1) $\mathcal{P}_n$ computes f from $p_1^{(n+c_1)},...,p_k^{(n+c_1)}$ to $q^{(n+c_1)}$ on interval $[0, 2^{2^n}-1]$ for some constant c_1 independent of n;
(2) the size of $\mathcal{P}_n$ is bounded by some polynomial of n.

To proof the lemma we notice that if the length of each input argument x_i is bounded by 2^n then the length of the tape of $\mathcal{M}$ is bounded by $ck2^n$ and its content can be represented by binary number of the length 2^{n+c_1} for some constant c_1. Therefore an instantaneous description of $\mathcal{M}$ of the form $w_l q_i a_j w_r$ can be represented by two $(n+c_1)$-ary predicates p_l and p_r for w_l and w_r, and a finite number 0-ary predicates for states q_i and tape symbols a_j. Then actions of $\mathcal{M}$ on the tape can be simulated by a standard sequence of arithmetical operations over the representations of instantaneous descriptions.

Let us fix a deterministic Turing machine $\mathcal{M}$ which works in time bounded by $2^{2^{p(n)}}$ for some polynomial $p(n)$, n being the length of an input word $x = a_{i_1}a_{i_2}...a_{i_n}$. We set $N = p(n)$.

We define by $\mathcal{M}$ and x such $DATALOG^u$-program $\mathcal{P}$, the IC Φ, and the disturbance δ that $\mathcal{M}$ accepts x iff $(<\mathcal{P}, \Phi>, \delta) \in HOM^T(DATALOG^u + IC_2)$.

Let $A = \{a_0, a_1, \ldots, a_m\}$ be the tape alphabet and $Q = \{q_0, \ldots, q_r\}$ be the state alphabet of $\mathcal{M}$ where q_0 is the initial state, $q_{r-1} =$ "no" is the rejecting state, and $q_r =$ "yes" is the accepting state. Following to S.Cook [2] we assume without lose of generality that the movement of the head of $\mathcal{M}$ can be devided by the following stages. At the 1-st stage the head of $\mathcal{M}$ marks the first cell and moves from the left to the right up to the first empty cell, marks it as "END" and returns to the first cell. At the stage i $(i \ge 1)$ the head starts in the first cell, then it moves from the left to the right up to the mark "END", shifts this mark to the right into the next empty cell, and returns to the first cell. So, the stage i takes $2(n+i)$ steps. The position of the head of $\mathcal{M}$ at any step t does not depend on x. Let us denote by $c(t)$ the function which given number of step t returns the number of cell in which the head is at the step t. Let $prev\_time(t) = max\{t' \mid t' < t \ \&\ c(t') = c(t)\}$ and $next\_time(t) = min\{t' \mid t < t' \ \&\ c(t') = c(t)\}$. Let the predicate $first\_time(t)$ is true iff the head attends cell $c(t)$ for the first time at the step t ($prev\_time(t)$ is undefined).

It is easy to check that all functions $c, prev\_time, next\_time,$ and $first\_time$ can be computed in linear space by Turing machines. So by lemma 2 they can be computed on interval $[0, 2^{2^N}]$ by $DATALOG^u$-programs of polynomial size which use predicates of arity $\leq 2N$.

The program $\mathcal{P}$ uses predicate $t^{(2N)}$ to represent the current number of steps of $\mathcal{M}$, and includes programs which compute the following auxiliary arithmetical operations and relations on t (we use t to denote the current number of steps representing by the facts of the form $t(i_1, ..., i_{2n})$ as well):

- $time^{+1}$ increases current t by 1;
- $time^{-1}$ decreases current t by 1;
- $first\_time$ succeeds if $first\_time(t)$ is true;
- $time_i$ $(0 \leq n)$ succeeds if $t = i$;
- $time_{gn}$ succeeds if $t > n$;
- $prev\_time$ computes $prev\_time(t)$;
- $next\_time$ computes $next\_time(t)$.

Let $AP = \{q_1^{(m_1)}, ..., q_r^{(m_r)}\}$ be the set of all auxiliary extensional predicates which are used in computations of these arithmetical operations and relations. We assume that no facts with predicates of AP are in DB state after any of these computations. Then $\mathbf{P}^e = \{t^{(2N)}, bad^{(0)}, good^{(0)}\} \cup AP$. Let update $delete\_all$ transforms every DB state over $\mathbf{P}^e$ in the empty one. The main part of $\mathcal{P}$ includes the following clauses.

(1) go :– $good$
(2) go :– $\neg bad$
(3) go :– $bad, time_0, step$
(4) $step$:– $head(yes, X)$, $delete\_all$, $insert(good)$
(5) $step$:– $time^{+1}, step$
(6) $head(1, i_k)$:– $first\_time$, $time_k$ $(1 \leq k \leq n)$
(7) $head(j, 0)$:– $first\_time$, $time_{gn}$, $time^{-1}$, $head(j', i')$, $time^{+1}$

for every pair (j', i') such that $\mathcal{M}$ includes instruction of the form $q_{j'}a_{i'} \to q_j a C$ for some $a \in A$ and $C \in \{-1, 0, 1\}$.

(8) $head(j, i)$:– $\neg first\_time$, $time^{-1}$, $head(j', i')$, $time^{+1}$, $prev\_time$, $head(j'', i'')$, $next\_time$

for every two pairs (j', i') and (j'', i'') such that $\mathcal{M}$ includes instructions of the form $q_{j'}a_{i'} \to q_j a' C'$ and $q_{j''}a_{i''} \to q'' a_i C''$ for some $a' \in A$, $q'' \in Q$ and $C', C'' \in \{-1, 0, 1\}$.

The following lemma asserts the main property of $head(j, i)$.

Lemma 3 *Let a DB state $\mathcal{E}$ represents a number s by $t^{(2N)}$. Then for each $q_j \in Q$ and $a_i \in A$ the computation of $head(j, i)$ on $\mathcal{E}$ succeeds iff on the step s of the computation $\mathcal{M}$ on input x the head of $\mathcal{M}$ visits in the state q_j a cell with the symbol a_i.*

Let IC $Phi = empty \vee good$, where formula $empty$ says that tere are no facts in DB state, and $\delta = (\{bad\}, \emptyset)$. Then due to the clauses (1) and (2) total homeostaticity of $(< \mathcal{P} \cup \{ :– go\}, \Phi >, \delta)$ depends only of its behavior on admissible empty state when disturbance adds the fact bad. And the only way to get into admissible state is to finish the computation of update go by the clause (4). Then the following assertion finishes the proof of lower bound.

Lemma 4 $\{bad\} \overset{go}{\vdash} \{good\}$ *iff* $\mathcal{M}$ *accepts input* x *in time* $2^{2^{|x|}}$.

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 8 *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* $D_1/_\equiv \subseteq D_2/_\equiv$. $D_1 \Leftrightarrows D_2$ *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* $\mathcal{E}_1 \Leftrightarrows \mathcal{E}_2$ *and* $\mathcal{E}_1 \Re \mathcal{E}'_1$, *there exists such a DB state* $\mathcal{E}'_2$ *that* $\mathcal{E}'_1 \Leftrightarrows \mathcal{E}'_2$ *and* $\mathcal{E}_2 \Re \mathcal{E}'_2$.

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

We can therefore factorize DDBs as well.

Definition 9 *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* $\overset{a_i}{\vdash_\mathcal{P}}$ *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* $\mathcal{E}$ $c(\delta_\equiv, \mathcal{E}) = \{(D^+, D^-) | D^+ \Leftarrow \Delta^+, D^- \Leftarrow \Delta^-\}$. *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 *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, thel* $\mid \tilde{\delta} \mid \leq 2^{pol(|\mathcal{B}|+|\mathcal{E}|+|\delta|)}$.

Lemma 5 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, $\tilde{\delta}$ 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.

6 Conclusion

The proposed concept of total homeostatic behavior of DDBs substantially generalizes the following their property: "for any given external update violating the IC there exists a reaction of the data base, which restores the IC". However, it has much broader scope and can as well be applied to the analysis of behavior of discrete dynamic systems with complex states 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. Among various possible types of stable behavior we choose here only homeostaticity, because of its close relation to data base scenarios. Meanwhile, in other application domains we encounter other types of stable behavior e.g. stable or perspective one, formalized and explored in our previous papers [4, 5, 6]. We consider rather broad classes of DDBs. So not surprisingly the complexity of the homeostaticity property in these classes is sometimes very complex. But in real applications the intensional description of system behavior is formed from great variety of primitive conditions. E.g., in active data bases the triggered rules are simple productions. For such primitive DDBs our results guarantee quite tractable upper complexity bounds. In any case very helpful interactive environments can be created on the ground of proposed concepts which support experimental analysis of behavior of discrete dynamic systems in a broad class.

References

[1] Apt, K.R., Blair, H. and Walker A., *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] Cook S.A., em Characterization of push-down machines in terms of time-bounded computers. J. of ACM, 18, N 1, 4-18, 1971.

[3] Dayal, U., Hanson,E., and Widom, J., *Active database systems.* In W. Kim, editor, *Modern Database Systems.* 436-456, Addison Wesley, 1995.

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

[5] Dekhtyar, M.I., Dikovsky, A.Ja., *On Homeostatic Behavior of Dynamic Deductive Data Bases.* Perspectives of Syst. Informatics. In *Extended Abstracts of the Andrei Ershov 2nd Intern. Memorial Conf.* Novosibirsk, Russia, 196-201, 1996. (The full draft is to appear in LNCS).

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

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

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

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

[10] Halfeld Ferrari Alves, M., Laurent, D., Spyratos, N. *Update rules in Datalog programs.* Rapport de Recherche n. 1024, 01 / 1996, Université de Paris-Sud, Centre d'Orsay, LRI.

[11] Katsuno, H., Mendelzon, A. O., *Propositional knowledge base revision and minimal change.* Artificial Intelligence, vol. 52, 253-294, 1991.

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

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