On Homeostatic Behavior of Dynamic Deductive Data Bases. [1]

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

Alexander Ja. Dikovsky
Keldysh Inst. for Appl. Math.
4 Miusskaya sq.
Moscow, Russia, 125047
dikovsky@applmat.msk.su

Abstract

We define a new property of steady behavior of dynamic deductive data bases interacting with their active medium. This property is in fact the homeostaticity, i.e. the ability of the database to restore its integrity constraints after ruinous disturbances of its medium. The formal model we use is based on a new concept, that of bounded disturbance of an active medium. We explore the computational complexity of this property in various classes of dynamic deductive data bases.

1 Introduction

This work treats the behavior of discrete dynamic systems in the frame of logic programming. It is in the line of our previous papers [6–8] where we have proposed a notion of a *stable* behavior of systems of this kind. In these papers the discrete dynamic systems are represented by dynamic deductive databases (DDDBs), and their states are represented by the sets of extensional atoms defined by the DDDBs, i.e. by databases (DBs). At every fixed moment the DB state can be admissible or not, the criterion of admissibility being implemented by the integrity constraints (IC) of the DDDB. The behavior of a discrete dynamic system is represented as a process of interaction of the DDDB with its active medium embodied by an operator of external disturbances on DB states. So, if not to say of the external disturbances operator, our treatment of discrete dynamic systems fits the traditional frame of deductive data bases as e.g. in [11]. Rather traditionnally (cf. [14]) we presume that the extensional predicates are not defined by the clauses of the intensional logic program underlying the DDDB, though they can be used and updated. As usual, the elementary updates used in the bodies of intensional clauses are either $insert(L, DB)$, or $delete(L, DB)$, and apply to extensional facts. This form of dynamic logic programs exactly fits our purposes, because it makes them represent intensional nondeterministic operators on extensional DB states. The intensional operator is induced by a given (thus predefined) goal, which therefore, can be regarded as a complex (nonelementary) update of DB states. An initial DB state being given, the intensional operator transforms it nondeterministically and nonmonotonically into a new

[1] This work was sponsored by INTAS (Grant 94-2412) and by the Russian Fundamental Studies Foundation (Grant 95-01-01321).

DB state, whose admissibility is guarded by the IC.

In the recent decade extensive research was done in the field of deductive databases with updates. Most attention was focussed on update languages and on the analysis of their expressibility and complexity (cf. [1, 3, 14, 16]) as well as of declarative and operational semantics of updates (cf. [4, 9, 17]), the latter being closely connected with the analysis of revision and updates in propositional knowledge bases (cf. [10, 13]). The central problem in this field is the determination of the effect of updates and validation of their consistency with IC. There are some interesting recent publications which present methods of transforming an updated database into an admissible one through its revision via the IC [15], or of updating admissible DB states in a manner so as to preserve their admissibility [12]. In fact both methods serve for strategies which guarantee steadyness of the evolution of DB states with respect to updates. Our approach to steady behavior is very different, and the difference in the approaches is determined by the difference in the implicit applications. Traditional data base applications face the consequences of data base updates produced by the users. One just cannot forsee the effect of such updates. Hence it is not sensible to measure or restrict them somehow. Updates violating IC are treated as abnormal. On the contrary, the modifications of states of a dynamic system are either produced by the system itself (*updates*) or by its active medium (*disturbancees*). One partner of the interaction may cause a violation of the IC, while the other one may restore it. Moreover, normally the effects of possible disturbances of the medium are somehow taken into account by the system control. What cannot be forseen, is the moment and the system state in which the disturbance acts. Nevertheless, the system may be provably capable of recovering from ruinous external disturbances whose effect on the states is somehow *uniformly bounded.* This is how we treat in this paper the *homeostatic behavior* of dynamic systems. The notion of stable behavior introduced in [6–8] is dual with respect to this. There are numerous classes of important applications calling for such an approach to updates. A typical example is any venture rendering services upon orders. Its current state meets the "integrity constraints" if all the incoming orders are satisfied. The receipt of a new order **does not depend on rules of functioning of the venture,** and therefore should be considered as a disturbance of its active medium. This venture's behavior is homeostatic if it manages to fulfil all its orders within available resources and to resume the consumed resources spending partly its income from the orders. Another example is a system of control of some complex hardware, which has particular strategies of recovery from certain external influences at the condition that they are within definite constraints.

So, a DDDB $\mathcal{B}$ simulating the behavior of a dynamic system $\mathcal{S}$ represents states of $\mathcal{S}$ by its DB states. An action of $\mathcal{S}$ in a state represented by a DB state $\mathcal{E}_1$ is represented by a predefined in $\mathcal{B}$ update $\mathcal{E}_1 \overset{G}{\vdash} \mathcal{E}_2$ of $\mathcal{E}_1$ which is induced by certain goal $\leftarrow G$ and implemented by a successful refutation of $\leftarrow G$ starting in $\mathcal{E}_1$ and resulting in $\mathcal{E}_2$. In the presence of nonmonotonic means this operator certainly should be properly precised, and this will be done below. A disturbance in a state represented by $\mathcal{E}_1$ is a nondeterministic

transformation $\mathcal{E}_1 \xrightarrow{d} \mathcal{E}_2$ which directly and unconditionnally adds and/or deletes some facts to/from $\mathcal{E}_1$ to obtain $\mathcal{E}_2$. The pincipal difference between the updates and the disturbances is determined by the asymmetry of their roles with respect to the IC. A certain way to express upper bounds on the size of the disturbances should be introduced. Some uniform upper bound ϱ being fixed, we say that $\mathcal{B}$ behaves ϱ*−homeostatically* in a state $\mathcal{E}$ satisfying its IC if after any disturbance $\mathcal{E} \xrightarrow{d} \mathcal{E}^*$ transforming $\mathcal{E}$ to a DB state $\mathcal{E}^*$ and bounded by ϱ there always exists a restoring update of $\mathcal{B}$ $\mathcal{E}^* \overset{G}{\vdash} \mathcal{E}'$ resulting in the state $\mathcal{E}'$ in which the IC is already satisfied and in which $\mathcal{B}$ is ϱ-homeostatic again. One may think about the space of all *trajectories* of $\mathcal{B}$, i.e. the sequences of form

$$\sigma: \mathcal{E}_0 \xrightarrow{d_1} \mathcal{E}_1^* \overset{G_1}{\vdash} \mathcal{E}_1 \xrightarrow{d_2} \mathcal{E}_2^* \overset{G_2}{\vdash} \ldots.$$

Then $\mathcal{B}$ is ϱ-homeostatic in $\mathcal{E}$ if in the subspace of all trajectories starting in this DB state there is a $\forall\exists-$subspace of trajectories in which all DB states $\mathcal{E}_i$ are *admissible,* and all disturbances are bounded by ϱ. The ϱ-bounded homeostaticity is much weaker than just have a restoring update for any disturbance bounded by ϱ, which is also quite reasonable though too restrictive to fit all the applications. This stronger property is less complex and is studied elsewhere.

In this paper we explore the computational complexity of the ϱ-bounded homeostaticity. The totality problem of partially recursive functions can be easily reduced to this property in the class of all DDDBs. However it is solvable in various classes of DDDBs via reasonable classes of IC of interest for real applications. E.g., in the nonrecursive DATALOG case where DDDBs are structureless production systems this problem is deterministic-double-exponential-time-complete. If DDDBs are ground (i.e. variableless) production systems, then this problem is deterministic-exponential-time-complete. In some more narrow classes of DDDBs the complexity of the ϱ-bounded homeostacity problem is quite workable. For example, in the class of ground production systems which never delete facts from states and under monotonic integrity constraints the problem is solved in polynomial time. The central result of this paper shows that (similarly to ϱ-bounded stability) under the restriction that the DATALOG or ground DDDBs are stratified with respect to elementary updates the complexity of the homeostaticity problem is the same as that of nonrecursive productional DDDBs.

2 Preliminaries and Main Definitions

We consider logic programs with updates in a first-order language $\mathbf{L}$. Along with the set of predicate symbols $\mathbf{P}$ of $\mathbf{L}$ we admit in general the set $\mathbf{F}$ of its functors. To obtain a natural semantics of updates we, similarly to [14], split $\mathbf{P}$ into two disjoint parts $\mathbf{P}^e$ (*extensional* predicates) and $\mathbf{P}^i$ (*intensional* predicates). Accordingly, the Herbrand base $\mathbf{B}$ of $\mathbf{L}$ splits into $\mathbf{B}^e = \{p(t_1, \ldots, t_k) \mid t_i \in \mathbf{H}, p \in \mathbf{P}^e\}$ and $\mathbf{B}^i = \{q(s_1, \ldots, s_l) \mid s_i \in \mathbf{H}, q \in \mathbf{P}^i\}$, where $\mathbf{H}$ is the Herbrand universe of $\mathbf{L}$. *DB states* are subsets of $\mathbf{B}^e$. The heads of clauses of logic programs are always intensional atoms. The bodies of clauses

can include two kinds of elementary database updates: $+A(\ldots)$, $-A(\ldots)$, where A **is an extensional atom**. We also allow negative **extensional atoms** in the clause bodies. To save the size of the programs, the bodies may include the disjunction " " as well as the update operator *change(A,B)* in place of the combination *delete(A), insert(B).*

We rely on the traditional SLD-refutation operational semantics of logic programs with the following specific features. The Prolog like leftmost selection computation rule is used with safe calls of elementary updates (i.e. all variables in the arguments of updates of the form $+A(\ldots)$ or $-A(\ldots)$ should be bound by ground terms by the moment of their call). Negation is treated as finite failure, which means in our very restricted situation that a negated atom in the current state cannot be unified to any fact in the current DB state. Successful refutations have a side effect of transforming initial DB states into new DB states. Any logic program $\mathcal{I}$ with updates can thus be naturally associated with the following relation $\mathcal{E} \vdash^{G}_{\mathcal{I}} \mathcal{E}'$ on DB states.

Definition 1 *Let $\mathcal{I}$ be a logic program, $\mathcal{E}, \mathcal{E}'$ be two DB states and $\leftarrow G$ be a goal. A refutation of $\mathcal{I} \cup \mathcal{E} \cup \{\leftarrow G\}$ is a finite sequence $S = s^1, s^2, \ldots, s^n$ of triplets (computation states) of the form $s^i = (\mathcal{E}_i, G^i, \sigma^i)$, in which $\mathcal{E}_i$ is a DB state, G^i is a goal and σ^i is a substitution, $\mathcal{E}_1 = \mathcal{E}$, $G^1 = G$, σ^1 is the identity substitution ε, G^n is the empty goal $\Box$ and each step (s^i, s^{i+1}), $i < n$, is of one of the following types. Let $G^i = L^1, L^2, \ldots, L^k$.*

(1) When L^1 is an atom, then for some variant of a clause (possibly a unit one) $H \leftarrow B^1, \ldots, B^r$ $(r \geq 0)$ and for MGU θ of $L^1 \circ \sigma^i$ and H the equalities $G^{i+1} = B^1, \ldots, B^r, L^2, \ldots, L^k$, $\sigma^{i+1} = \sigma^i \circ \theta$ and $\mathcal{E}_{i+1} = \mathcal{E}_i$ hold as usually.

(2) When $L^1 = \neg b(\bar{t})$ for some $b \in \mathcal{P}^e$ and $b(\bar{t}) \circ \sigma^i$ cannot be unified to any fact in $\mathcal{E}_i$, then the equalities $G^{i+1} = L^2, \ldots, L^k$, $\mathcal{E}_{i+1} = \mathcal{E}_i$ and $\sigma^{i+1} = \sigma^i$ hold.

(3) When $L^1 = +A(\ldots)$ and $A \circ \sigma^i \in \mathbf{B}^e$ then the equalities $G^{i+1} = L^2, \ldots, L^k$, $\mathcal{E}_{i+1} = \mathcal{E}_i \cup \{A \circ \sigma^i\}$ and $\sigma^{i+1} = \sigma^i$ hold.

(4) When $L^1 = -A(\ldots)$ and $A \circ \sigma^i \in \mathbf{B}^e$ then the equalities $G^{i+1} = L^2, \ldots, L^k$, $\mathcal{E}_{i+1} = \mathcal{E}_i \setminus \{A \circ \sigma^i\}$ and $\sigma^{i+1} = \sigma^i$ hold.

When such a refutation exists, we write the usual $\mathcal{I} \cup \mathcal{E} \cup \{\leftarrow G\} \Longrightarrow \Box$. If in addition $\mathcal{E}_n = \mathcal{E}'$, then we say that the program $\mathcal{I}$ with the goal $\{\leftarrow G\}$ transforms $\mathcal{E}$ into $\mathcal{E}'$, and denote this by $\mathcal{E} \vdash^{G}_{\mathcal{I}} \mathcal{E}'$. We will indicate by $\mathcal{E} \vdash^{p}_{\mathcal{I}} \mathcal{E}'$ specifically the one step refutations where a unique clause b is applied.

As is readily seen from this definition, we consider each goal for a logic program with updates as a pre-defined nondeterministic transformation of DB states (an *update*).

In many application databases the data can be naturally stratified into classes where specific values of certain attributes are purely informational, i.e. their change does not affect operational properties of the data base. The data dif-

fering only in such "insignificant" features can be regarded as equivalent. The partitioning of data into classes of data equivalence sometimes permits reduction of a large and potentially infinite application domain to a bounded and visible one. For example, so as to investigate the cause of the queues in a library (i.e. the loss of the homeostaticity) on the basis of dynamic records of services during one month, one should abstract from dates, particular identities of clients, the biblographic data of the ordered items, etc. As a result any two records of the same service, falling into the same interval of time during a day become equivalent.

It is clear that every equivalence $\equiv$ on $\mathbf{H}$ can be extended naturally onto $\mathbf{B}^e$ by $\vartheta(t^1,\ldots,t^n) \equiv \vartheta(t'^1,\ldots,t'^n)$ iff $t^i \equiv t'^i$ for all $1 \leq i \leq n$.

Definition 2 *For every* $D^1, D^2 \subseteq \mathbf{B}^e$ *we set* $D^1 \overset{\equiv}{<} D^2$ *if* $D^1/\equiv \subseteq D^2/\equiv$. $D^1 \overset{\equiv}{<>} D^2$ *if* $D^1 \overset{\equiv}{<} D^2$ *and* $D^2 \overset{\equiv}{<} D^1$.

A logic program $\mathcal{I} \cup \{ :- G \}$ *is* compatible with an equivalence $\equiv$ *if for any three states* $\mathcal{E}_1, \mathcal{E}'_1, \mathcal{E}_2 \subseteq \mathbf{B}^e$ *such that* $\mathcal{E}_1 \overset{\equiv}{<>} \mathcal{E}_2$ *and* $\mathcal{E}_1 \overset{G}{\vdash}_{\mathcal{I}} \mathcal{E}'_1$ *there exists such a state* $\mathcal{E}'_2$ *that* $\mathcal{E}'_1 \overset{\equiv}{<>} \mathcal{E}'_2$ *and* $\mathcal{E}_2 \vdash_{\mathcal{I}} \mathcal{E}'_2$.

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

The deductive data bases we consider are always compatible with some data equivalence (which might be the equality either).

Definition 3 *A* dynamic deductive data base (DDDB) *in language* $\mathbf{L}$ *is a system*

$$\mathcal{B} = < \mathcal{I} \cup \{ :- G^1, \ldots, :- G^n \}, \Phi, \equiv >$$

where:

- $\mathcal{I}$ *is a logic program with updates in* $\mathbf{L}$,
- *all goals* $G^i, i = 1, \ldots, n$, *are ground, we call them* updates,
- Φ *is some closed 1-st order formula in extensional signature defining* integrity constraints (IC),
- $\equiv$ *is an equivalence on* $\mathbf{B}^e$ *(we call it a* data equivalence*), and logic programs* $\mathcal{I} \cup \{ :- G^i \}, i = 1, \ldots, n$, *and the IC* Φ *are compatible with* $\equiv$.

Now we turn to our main definition. Its core concept is the notion of trajectory with disturbances which makes behavior of a DDDB reactive towards the effect of its medium.

Definition 4 *For any two disjoint sets* $\mathcal{D}^+, \mathcal{D}^- \subseteq \mathbf{B}^e$, *we define the* $(\mathcal{D}^+, \mathcal{D}^-)$-disturbance *as the relation on DB states* $\mathcal{E}_1 \overset{\mathcal{D}^+,\mathcal{D}^-}{\longrightarrow} \mathcal{E}_2$ *such that* $\mathcal{E}_1, \mathcal{E}_2 \subseteq \mathbf{B}^e$ *and* $\mathcal{E}_2 = (\mathcal{E}_1 \cup \mathcal{D}^+) \setminus \mathcal{D}^-$.

Let $\mathcal{B} = < \mathcal{I} \cup \{ :- G^1, \ldots, :- G^n \}, \Phi, \equiv >$ *be a DDDB. We define a* trajectory *of* $\mathcal{B}$ *as any finite or infinite sequence of the form*

$$\mathfrak{T}: \ \mathcal{E}_0 \overset{\mathcal{D}^+_1,\mathcal{D}^-_1}{\longrightarrow} \mathcal{E}^*_1 \overset{G_{i_1}}{\vdash}_{\mathcal{I}} \mathcal{E}_1 \overset{\mathcal{D}^+_2,\mathcal{D}^-_2}{\longrightarrow} \mathcal{E}^*_2 \overset{G_{i_2}}{\vdash}_{\mathcal{I}} \mathcal{E}_2 \ldots .$$

Let $\varrho = < \mathcal{D}^+, \mathcal{D}^- >$ *with finite* $\mathcal{D}^+, \mathcal{D}^- \subseteq \mathbf{B}^e$. *Then* $\mathfrak{T}$ *is* ϱ−bounded *if for all* $k \geq 1$ $\mathcal{D}^+_k \overset{\equiv}{<} \mathcal{D}^+$ and $\mathcal{D}^-_k \overset{\equiv}{<} \mathcal{D}^-$.

A trajectory ω *is* homeostatic *if every its DB state* $\mathcal{E}_i$, $i = 0, 1, 2, \ldots$ *satisfies the IC* Φ, *i.e.* $\mathcal{E}_i \models \Phi$.

Homeostaticity of a trajectory means that all IC violations caused by disturbances along the trajectory are compensated by subsequent updates. It is clear that the set of all ρ−bounded trajectories starting in a fixed state $\mathcal{E}$ forms a tree T of infinite in general height, in which all nodes of even depth have the fan-out bounded by $2^{|\delta|}$. By its $\forall\exists$−subtree we mean as usual any its subtree whose every even-depth nonleaf node includes all its sons in T, and every odd-depth nonleaf node includes exactly one of its sons in T.

Definition 5 *Let* $\mathcal{B}$ *be a DDDB and* $\rho = <\mathcal{D}^+, \mathcal{D}^->$ *with finite* $\mathcal{D}^+, \mathcal{D}^- \subseteq \mathbf{B}^e$. *A DDDB* $\mathcal{B}$ *is* ρ-homeostatic in a DB state $\mathcal{E}$ *if in the tree of all its* ρ*-bounded trajectories starting in* $\mathcal{E}$ *there is a* $\forall\exists$−*subtree in which all branches are infinite homeostatic trajectories.*

Example 1. Consider the following toy DDDB $\mathcal{B}_1$ describing patient's health condition in terms of three propositional extensional predicates $disease$, which may appear as the external medium disturbance, $medicine$, which represents the availability of necessary remedy, and $immunity$, which emerges as a result of curing the disease. The intensional logic program consists of three updates:

upd_1 :− $disease, medicine, delete(disease), delete(medicine), insert(immunity)$.

upd_2 :− $disease, immunity, delete(disease), delete(immunity), insert(medicine)$.

upd_3 :− $\neg disease$.

The IC $\Phi^1 = \neg disease$ describes admissible states.
This DDDB is $(\{disease\}, \emptyset$

homeostatic in any of states $\{a, \neg b\}$, $\{\neg a, \neg b\}$, $\{a, b\}$, $\{\neg a, b\}$, if at most one fact from Q is added to a state on every disturbance step. But it is not Q-homeostatic in any state without this restriction on disturbance size, because its rules cannot cope with two diseases at one time.

3 A Classification of DDDBs

The problems we are dealing with in this work include as a subproblem the satisfiability of IC, i.e. the validity problem of a 1-st order formula in a finite model (a DB state). This problem is evidently solvable in polynomial space. On the other hand there are classes of applications in which integrity constraints are rather primitive (expert systems, planning, etc.). Therefore we consider below several classes of DDDBs whose IC Φ satisfy one of the following conditions:

(IC0) Φ is ground and does not contain negation (is monotonic)
(IC1) Φ is ground
(IC2) Φ is an existentially closed formula in prenex form.

We impose several workable conditions on intensional parts of DDDBs and analyze their impact on computational complexity of properties of their behavior.

Definition 6 *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 variables or constants. We call $\mathcal{P}$* expanding *if the update $assert$ is not used in its clauses. 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 $q :- L^1, \ldots, L^k, A^1, \ldots, A^m$ where each L^i is an extensional literal and each A^j is an elementary update.*

The term "production" is borrowed from AI. The clause $q :- L^1, \ldots, L^k, A^1, \ldots, A^m$ corresponds there to the conditional transformation of the form $CONDITION \Rightarrow ACTION$ where $CONDITION$ is the conjunction of the literals L^i and $ACTION$ is the sequence of updates $A^1, \ldots, A^m$. Similar rules have been used e.g. in the transaction language DL in [1].

By $DDDB$ we denote the set of all DDDBs with productional intensional programs $\mathcal{I}$. Within this class we introduce several subclasses:

- $DDDB^F$ is the subclass of all DDDBs in $DDDB$ with flat intensional parts
- $DDDB^G$ is the subclass of all DDDBs in $DDDB$ with ground intensional parts
- $DDDB^{G-}$ is the subclass of all DDDBs in $DDDB^G$ with expanding intensional parts
- $DDDB^{G+}$ is the subclass of all DDDBs in $DDDB^{G-}$ with positive intentional parts (hence intensional parts of programs in $DDDB^{G+}$ do not use negation and the update $assert$).

Productional DDDBs are not recursive. We impose on recursion a constraint analogous in a sense to the stratifiability property in [2].

Definition 7 *Let* $\mathcal{P}$ *be a logic program. We say that a predicate* p refers to *a predicate* q *if there is a clause defining* p *in* $\mathcal{P}$ *with* q *in its body. 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 of the relation "depend on" are called* cliques. $\mathcal{P}$ *is called* dynamically stratified (d-stratified) *if in any of its clauses*

$$p(\bar{t}) \leftarrow b^1(\bar{t}^1), \ldots, b^i(\bar{t}^i), q(\bar{s}), b^{i+1}(\bar{t}^{i+1}), \ldots, b^r(\bar{t}^r)$$

in which q *is in the clique of* p*, all predicates* $b^1, \ldots, b^i, b^{i+1}, \ldots, b^r$ *are* stationary, *i.e. do not depend on elementary updates.*

The main point of this definition is that DB updates are available only at the steps where a clique is changed. Imposing this constraint we introduce the following superclasses of DATALOG and DATALOG$^{\neg}$. **GDS** and **FDS** will denote the classes of all DDDBs having respectively ground or flat d-stratified intensional parts and IC2-type integrity constraints.

4 Complexity of Homeostaticity Problem

In this section we estimate the complexity of the problems of homeostaticity in the above introduced classes of DDDBs. For a class of DDDBs **P** we consider the problem:

$$\text{HOM}(\mathbf{P}) = \{(\mathcal{B}, \mathcal{Q}, \mathcal{E}) \mid \mathcal{B} \in \mathbf{P} \text{ is homeostatic w.r.t. } \mathcal{Q} \text{ and } \mathcal{E}\}.$$

Of course this problem is undecidable in the class of all DDDBs. We use the following notation for complexity classes. Let EXPSPACE denote the class of all problems computable in polynomial space by the alternating Turing machines (it is well known [5] that this class is equal to the class of problems computable in exponential time by deterministic Turing machines). By $\text{EXPSPACE}(2^{poly})$ we denote the class of all problems computable by alternating Turing machines in space bounded by $2^{p(n)}$ for some polynomial p.

The following theorem establishes the complexity of homeostaticity for productional DDDBs.

Theorem 1 *(1) The problem* $\text{HOM}(\text{DATALOG}^{\neg +})$ *w.r.t.* IC *0-constraints is solvable in polynomial time.*

(2) The problem $\text{HOM}(\text{DATALOG}^{\neg +})$ *w.r.t.* IC *1-constraints is hard and the problem* $\text{HOM}(\text{DATALOG}^{\neg})$ *w.r.t.* IC *2-constraints belongs to* EXPSPACE*-*EXPSPACE*.*

(3) The problem $\text{HOM}(\text{DATALOG}^{\neg -})$ *is* EXPSPACE*-complete when* $\mathcal{D}^- \subseteq \mathcal{D}^+$ *holds for* $\mathcal{Q} = <\mathcal{D}^+, \mathcal{D}^->$*.*

(4) The problem $\text{HOM}(\text{DATALOG})$ *is* $\text{EXPSPACE}(2^{poly})$*-complete.*

(5) The problem $\text{HOM}(\text{DATALOG}^{\neg})$ *is undecidable.*

Proof. For the reasons of space we sketch briefly the upper bounds of points (1) and (3) and the lower bound of point (2).

(1) Let $\mathcal{B} = <\mathcal{P}, \Phi, \equiv> \in \text{DATALOG}^{\neg +}$, $\mathcal{Q} = <\mathcal{D}^+, \mathcal{D}^->$ and $\mathcal{E} \subset \mathbf{B}^e$. Then monotonicity of Φ and positivity of productions in $\mathcal{P}$ imply that

$(\mathcal{B}, \rho, \mathcal{E}) \in \exists\forall\text{-HORN-DATALOG}^{+}(\pi)$ iff there exists a production $\pi \in \mathcal{P}$ such that $(\mathcal{E} \setminus \mathcal{D}^{-}) \vdash \mathcal{E}_1$ and $\mathcal{E}_1 \models \Phi$. The latter can be checked in time $O(n^2)$ by some RAM under the standard coding of input of length n.

(2) *Lower bound.* We demonstrate the polynomial time reducibility of computations of an alternating Turing machine $\mathcal{M}$ with space bounded by a polynomial p to $\exists\forall\text{-HORN-DATALOG}(\pi)$. Let $\mathcal{M} = (Q^{\exists} \cup Q^{\forall}, \Sigma, P, q^0, \{q^a, q^r\})$ be such an alternating Turing machine, where $Q^{\exists}$ is the set of existential states, $Q^{\forall}$ is the set of universal states, $\Sigma = \{a^0, a^1, \ldots, a^m\}$ is the tape alphabet, P is the (non) instruction set, q^0 is the initial state, q^a is the accepting final state, and q^r is the rejecting final state.

Without loss of generality we suppose that $q^0, q^a, q^r \in Q^{\forall}$, $a^0 \in \Sigma$ is a blank symbol, and P does not contain more than two instructions with the same left side qa. We fix some order of the two instructions and call them respectively the first and the second instruction for qa. Besides, we presume that universal and existential states in computations of $\mathcal{M}$ alternate. Let $x = a^{i_1}, \ldots, a^{i_n}$ be an input word and $N = p(n)$ be the space bound. The extensional signature $\mathbf{P}^e$ of $\mathcal{B}$ consists of the following 0-ary predicates:

- $q^{\forall}, q^{\exists}$ for all $q^{\forall} \in Q^{\forall}$
- $q^{\exists}, q^{\exists}$ for all $q^{\exists} \in Q^{\exists}$
- a^{tj}, a^{tj} $(1 \le t \le N, 0 \le j \le m)$: the cell t contains the symbol a^j
- h^t, h^t $(1 \le t \le N)$: the head of $\mathcal{M}$ is against the cell t
- ρ, c, l, s: control predicates (ρ indicates a contradiction in a DB state, c indicates the situation where too many facts were deleted by a disturbances, l, s control the choice of the first/second instruction for the given left side).

For disturbances we introduce the upper bound $\rho = (\emptyset, \mathbf{P}^e \setminus \{\rho, c, q^a, q^r\})$. So disturbances can only delete some of the listed facts.

The program $\mathcal{P}$ includes the following groups of productions.

1) $q^{\exists} \& h^j \& a^{jk} \Rightarrow \text{insert}(q^{\forall}, h^{j+S}, a^{jv})$

for every $1 \le j \le N$ and every instruction of the form $q^{\exists} a^k \to q^{\forall} a^v S$, where $q^{\exists} \in Q^{\exists}$, $q^{\forall} \in Q^{\forall}$, and $S \in \{-1, 0, 1\}$ defines the head shift.

2) $q^{\forall} \& h^j \& a^{jv} \Rightarrow \text{insert}(q^{\forall}, h^j, a^{jv}, l, s)$,

for all $q^{\forall} \in Q^{\forall}$, $1 \le j \le N$, and $a^v \in \Sigma$.

3) $l \& q^{\forall} \& h^j \& a^{jk} \Rightarrow \text{insert}(q^{\exists}_1, h^{j+S_1}, a^{jv_1})$

for every left side $q^{\forall} a^k$, for its first instruction $q^{\forall} a^k \to q^{\exists}_1 a^{v_1} S^1$, and for all $1 \le j \le N$.

4) $s \& q^{\forall} \& h^j \& a^{jk} \Rightarrow \text{insert}(q^{\exists}_2, h^{j+S_2}, a^{jv_2})$

for every left side $q^{\forall} a^k$, for its second instruction $q^{\forall} a^k \to q^{\exists}_2 a^{v_2} S^2$, and for all $1 \le j \le N$.

5) $q^{\exists} \& h^j \& a^{jv} \Rightarrow \text{insert}(q^{\exists}, h^j, a^{jv})$,

for all $q^{\exists} \in Q^{\exists}$, $1 \le j \le N$, and $a^v \in \Sigma$.

6) $\text{fail} \Rightarrow \text{insert}(c)$.

7) $c \Rightarrow$.

8) $q^a \Rightarrow$.

9) $\text{pair\_of\_facts} \Rightarrow \text{insert}(\rho)$,

where pair_of_facts is any pair of facts which shouldn't be present simultane-

ously in a DB-state representing a configuration of $\mathcal{M}$, i.e. any pair of one of the forms $(\mathbb{D}\&\mathbb{D}')$, $(\mathbb{D}\&\hat{\mathbb{D}}')$ $(\mathbb{P}^i\&\mathbb{P}^j)$, $i \neq j$, $(\mathbb{P}^i\&\hat{\mathbb{P}}^j)$ $(\mathbb{Q}^{jk}\&\mathbb{Q}^{jl})$, $k \neq l$ $(\mathbb{Q}^{jk}\&\hat{\mathbb{Q}}^{j,l})$ $(\mathbb{D}\&\hat{\mathbb{P}}^j)$ $(\hat{\mathbb{D}}\&\mathbb{P}^j)$ $(\mathbb{D}\&\mathbb{Q}^{jk})$ $(\mathbb{P}^j\&\hat{\mathbb{Q}}^{ik})$ $(\hat{\mathbb{P}}^j\&\mathbb{Q}^{jk})$ $(\mathbb{T}\&\mathbb{S})$.

10) $\mathbb{R} \Rightarrow \mathbb{R}$.

Here $\mathbb{R}(\mathbb{T}^1,\ldots,\mathbb{T}^k)$ abbreviates $\mathbb{R}(\mathbb{T}^1),\ldots,\mathbb{R}(\mathbb{T}^k)$.

The IC Φ specifies the set of admissible DB states as those including $\mathbb{D}^a$, or those not representing configurations of $\mathcal{M}$:

$$\Phi = \mathbb{D}^a \vee \mathbb{R} \vee (\neg\mathbb{R} \,\&\, \neg\mathbb{D}^r) \vee (\neg\mathbb{T} \,\&\, \neg\mathbb{S} \,\&\, \mathbb{C}) \vee$$

$$(\neg(\bigvee_{q\in Q^\exists \cup Q^\forall} \mathbb{D}) \,\&\, \neg(\bigvee_{q\in Q^\exists \cup Q^\forall} \hat{\mathbb{D}}) \,\&\, \mathbb{C}) \vee (\neg(\bigvee_{1\le j\le N} \mathbb{P}^j) \,\&\, (\neg(\bigvee_{1\le j\le N} \hat{\mathbb{P}}^j) \,\&\, \mathbb{C}) \vee$$

$$\bigvee_{1\le j\le N} (\mathbb{P}^j \,\&\, \neg(\bigvee_{0\le k\le m} \mathbb{Q}^{jk}) \,\&\, \mathbb{C}) \vee \bigvee_{1\le j\le N} (\hat{\mathbb{P}}^j \,\&\, \neg(\bigvee_{0\le k\le m} \hat{\mathbb{Q}}^{jk}) \,\&\, \mathbb{C}).$$

The starting instantaneous description of $\mathcal{M}$ is represented by the initial DB state $\mathcal{E}_x = \{\mathbb{T}, \mathbb{S}, \mathbb{D}^1, \mathbb{P}^1, \mathbb{Q}^{1i_1}, \ldots, \mathbb{Q}^{ni_n}, \mathbb{Q}^{(n+1)0}, \ldots, \mathbb{Q}^{N0}\}$.

Every $\forall\exists-$computation tree of $\mathcal{M}$ on x can be simulated by $\forall\exists-$subtree of homeostatic trajectories of $\mathcal{B}$ starting in $\mathcal{E}_x$ and finishing in a DB state containing $\mathbb{D}^a$ if $\mathcal{M}$ accepts x, or in a DB state containing $\mathbb{D}^r$ if $\mathcal{M}$ rejects x. The lower bound follows from the following lemma.

Lemma 1 *$\mathcal{M}$ accepts x in space N iff $(\mathcal{B}, \mathcal{E}_x) \in \forall\exists\text{-HOM}(\text{PROG}^+)$.*

(3) *Upper bound.* Let $\mathcal{B} = <\mathcal{P}, \Phi, \equiv> \in \text{PROG}^-$, $\mathcal{E}_0 = <\mathbf{D}^+, \mathbf{D}^->$, $\mathbf{D}^- \subseteq \mathbf{D}^+$ and $\mathcal{E}_0 \subseteq \mathbf{B}^e$ be an initial state. One can prove that $(\mathcal{B}, \mathcal{E}_0) \in \forall\exists\text{-HOM}(\text{PROG}^-)$ iff in the tree $\mathbb{T}$ of all trajectories starting in $\mathcal{E}_0$ and k-bounded w.r.t the equality as the data equivalence there is a $\forall\exists$-subtree $\mathbb{T}^{\forall\exists}$ in which all branches are infinite homeostatic trajectories. Let $\mathcal{P}$ have k productions. For a production $\pi \in \mathcal{P}$ let $\mathbb{D}^\pi$ denote the set of all facts which emerge as the result of firing π, and let $\mathbb{D}^\pi_- = \mathbb{D}^\pi \cap \mathcal{D}^-$, and $\mathbb{D}^\pi_+ = \mathbb{D}^\pi \setminus \mathbb{D}^\pi_-$. The lemma to follow allows to reduce the infinite tree $\mathbb{T}^{\forall\exists}$ to some its finite and small subtree.

Lemma 2 *The premise of the theorem being true, the tree of all trajectories has a $\forall\exists-$subtree $\mathbb{T}^{\forall\exists}$ in which all branches are infinite homeostatic trajectories iff $\mathbb{T}$ has a finite $\forall\exists-$subtree $\mathbb{T}'$ such that*
(i) all branches of $\mathbb{T}'$ are homeostatic trajectories of the length $\le 2(k+1)$
(ii) every leaf $\mathbb{S}$ of $\mathbb{T}'$ has an odd depth (i.e. it is a $\exists-$node) and there exists a non leaf node $\mathbb{S}'$ in $\mathbb{T}'$ with the same DB state as that of $\mathbb{S}$.

Proof of the lemma. ($\Rightarrow$) So as to define $\mathbb{T}'$ we describe the rule of pruning long branches of $\mathbb{T}^{\forall\exists}$. Let $\mathbb{B}$ be a branch in $\mathbb{T}^{\forall\exists}$ longer than $2(k+1)$. Then there is a production π in $\mathbb{B}$ fired at least twice. Consider the first step i where π is fired in $\mathbb{B}$ for the second time:

$$\mathbb{B}: \ \mathcal{E}_0, \ldots, \mathcal{E}_i^* \overset{\pi}{\vdash} \mathcal{E}_i, \ldots, \mathcal{E}_{j-1} \overset{\mathcal{D}_j^+, \mathcal{D}_j^-}{\longrightarrow} \mathcal{E}_j^* \overset{\pi}{\vdash} \mathcal{E}_j \overset{\mathcal{D}_{j+1}^+, \mathcal{D}_{j+1}^-}{\longrightarrow} \mathcal{E}_{j+1}^*, \ldots$$

The rest of this branch after $\mathcal{E}^*_{j+1}$ must be pruned. As π is minimal, we have $i \leq n+1$. Then $\mathcal{E}^*_{j+1} = (\mathcal{E}^*_j \cup \mathcal{D}^+_{j+1}) \setminus \mathcal{D}^-_{j+1} = (((\mathcal{E}_{j-1} \cup \mathcal{D}^+_j) \setminus \mathcal{D}^-_j) \cup \mathcal{D}^+_{j+1}) \setminus \mathcal{D}^-_{j+1} = (\mathcal{E}_{j-1} \cup \pi^+) \setminus \pi^-$ for $\pi^+ = \mathcal{D}^+_j \cup \mathcal{D}^+_{j+1} \subseteq \mathcal{D}^+$ and $\pi^- = (\mathcal{D}^-_j \setminus (\mathcal{D}^+_{j+1} \cup \mathcal{D}^-_{j+1})) \subseteq \mathcal{D}^-$ (we use here the fact that π^+ is present in all the DB states which arise after the j-th step on which π^+ is fired). Therefore, there is a son of $\mathcal{E}_{j-1}$ in $T^{\forall\exists}$ which is equal to $\mathcal{E}^*_{j+1}$ and is not a leaf. Note that this argument holds for every branch in $T^{\forall\exists}$ containing the node $\mathcal{E}_j$. Therefore, the subtree T' resulting after pruning all branches in $T^{\forall\exists}$ satisfies the conditions (i) and (ii) of the lemma.

($\Leftarrow$) Let T' be a finite $\forall\exists$-subtree of T which satisfies the condition (ii). Then it can be unfolded into an infinite $\forall\exists$-subtree $T^{\forall\exists}$ of T whose branches are infinite and homeostatic. Let $T^1 = T'$ and suppose that a tree T^i is defined that meets (ii). To obtain T^{i+1} we choose in T^i some leaf σ' of the smallest depth with DB state $\mathcal{E}^*$ and find the corresponding non node σ' with the same DB state. Let τ' be the (unique) son of σ' in T^i and $\mathcal{E}'$ be the DB state of τ'. Then we add to T^i a new son of σ' with the DB state $\mathcal{E}'$. After this we add to T^i all of the sons of τ' in T. It is easy to check that T^{i+1} meets again the condition (ii) and that the tree $T^{\forall\exists} = \bigcup_{i \geq 1} T^i$ is a $\forall\exists$-subtree of T whose branches are infinite and homeostatic.

To finish the proof we note that every DB state in T has the size bounded by the length of the input, that every arrow in T can be generated in polynomial time, and that the IC can be checked in polynomial space for every DB state. Then the existence of a $\forall\exists$-subtree of T of depth $2(n+1)$ with homeostatic branches can be checked in polynomial space using the standard algorithm of the simulation of ALTERNATING in DETERMINISTIC [5]. □

The following theorem which we cite without proof shows that the stratified recursion does not increase the complexity of the homeostaticity problem.

Theorem 2 *(1) The problem* HOMEOSTATICITY(DDS) *is* EXPTIME*-complete.*
(2) The problem HOMEOSTATICITY(SDDS) *is* EXPTIME(2^{poly})*-complete.*

5 Conclusion

Formulation of the property of homeostatic behavior in logical terms is a new resource for mathematical analysis of steady behavior of discrete dynamic systems. It might be more workable than the classical analytical definitions in the situations where the systems are characterized by multiple parameters and the premise of continuity of states changes fails. We should note that the high lower bounds of complexity we establish do not show that the notion of homeostaticity we have introduced is not workable. First of all there are classes of DDDBs of interest for applications where its complexity is polynomial. Besides this the exponential time and space upper bounds algorithms we propose are simple generate-check nondeterministic algorithms. In a specific domain they might possibly be effectively implemented. This makes us look forward to development of an interactive environment for computer aided simulation and analysis of complex discrete dynamic systems behavior.

References

1. Abiteboul, S., Vianu, V., *Datalog extensions for database queries and updates,* I.N.R.I.A. Technical Report No. 900, 1988.
2. 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.
3. Bonner, A.J., *Hypothetical Datalog: complexity and expressibility.* Theoretical Computer Science, 76, 3-51, 1990.
4. Bonner,A.G., Kifer, M., *Transaction logic programing,* In *Proc. of the Tenth Intern. Conf. on Logic Programming.* The MIT Press, 257-279, 1993.
5. Chandra, A.K., Kozen, D.C., Stockmeyer, L.J., *Alternation.* J. Ass. Comput. Mach., v.28, n.1, 114-133, 1981.
6. Dekhtyar, M.I., Dikovsky, A.Ja., *Dynamic deductive databases.* Izvestiya of Russian Acad. of Sci. Technical Cybernetics, n.5, 1994, (Russ).
7. 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.
8. Dekhtyar, M.I., Dikovsky, A.Ja., *Properties of Steady Behavior of Dynamic Deductive Data Bases. Part I. $\exists\exists$-stability and Promise.* Technical rept. 95-07, Universite Paris XII-Val de Marne, Septembre 1995, 1-26.
9. Dung, P.M., Representing actions in logic programming and its application in database updates. In *Proc. of the Tenth Intern. Conf. on Logic Programming.* The MIT Press, 222-238, 1993.
10. Eiter, T., Gottlob, G., *On the complexity of propositional knowledge base revision, updates, and counterfactuals.* Artificial Intelligence, vol. 57, 227-270, 1992.
11. Gallaire, H., Minker, J., Nicolas, J.-M., *Logic and databases: a deductive approach.* ACM Computing Surveys, vol. 16, n.2, 153-185, 1984.
12. Halfeld Ferrari Alves, M., Laurent, D., Spyratos, N., Stamate, D., *Update rules and revision programs.* Rapport de Recherche n. 1010, 12 / 1995, Université de Paris-Sud, Centre d'Orsay, LRI.
13. Katsuno, H., Mendelzon, A. O., *Propositional knowledge base revision and minimal change.* Artificial Intelligence, vol. 52, 253-294, 1991.
14. Manchanda, S., Warren, D.S., *A logic-based language for database updates.* In J. Minker, editor, *Foundations of Deductive Databases and Logic Programming,* Morgan-Kaufmann, Los Altos, CA, 363-394, 1988.
15. Marek, V.W., Truszciński, M. *Revision programming, database updates and integrity constraints.* In *International Conference on Data Base theory,* ICDT, LNCS n. 893.
16. Naqvi, S., Krishnamurthy, R., *Database updates in logic programming.* In *ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, 251-262, 1988.
17. Sadri, F., Kowalski, R., *A theorem proving approach to database integrity.* In J. Minker, editor, *Foundations of Deductive Databases and Logic Programming,* Morgan-Kaufmann, Los Altos, CA, 313-362, 1988.