# Using Agents for Concurrent Querying of Web-like Databases via a Hyper-Set-Theoretic Approach

Vladimir Sazonov

Department of Computer Science
University of Liverpool, Liverpool L69 7ZF, U.K.
http://www.csc.liv.ac.uk/~sazonov
V.Sazonov@csc.liv.ac.uk
Tel: (+44) 0151 794 6792
Fax: (+44) 0151 794 3715

**Abstract.** The aim of this paper is to present a brief outline of a further step in the ongoing work concerning the hyper-set-theoretic approach to (unstructured) distributed Web-like databases and corresponding query language $\Delta$. The novel idea in this paper consists in using dynamically created mobile agents (processes) for more efficient querying such databases by exploiting concurrently distributed computational resources, potentially over the whole Internet. A fragment of a calculus of querying agents based on $\Delta$ is presented.

## 1 Introduction

Querying and searching the World-Wide Web (WWW) or, more generally, *unstructured* or *Web-like Databases (WDB)* is one of the most important contemporary information processing tasks. There are several search engines such as Alta Vista, Lycos, etc., but their search can hardly be characterised as "goal-directed" and always "up to date". Also it is desirable not only to be able to find a list of Web-pages of potential interest, but to ask more complex queries allowing, additionally, reorganisation of Web data, as required. That is, the answer to a query should constitute a number of possibly newly created hyper-linked pages (re) constructed from some pages existing somewhere in WDB — very much in the same way as in a relational database. A new relation/WDB-page(s) (the answer to a query) is the result of reconstructing existing ones in the database. In fact, our approach to WDBs is *a natural generalisation of the traditional relational databases* and differs essentially from the other known considerations of semi-structured databases, being actually *(hyper) set-theoretic* one.

The starting point for this work was the characterisation of PTIME computability in terms of recursion over finite structures obtained by the author [32] and independently by N. Immerman, A. Livchak, M. Vardi and Y. Gurevich [18, 20, 26, 42, 18]. It should be mentioned, of course, the seminal previous work of R. Fagin [14] on describing NPTIME in terms of existential second-order logic. Such results were also considered in a *framework of an abstract* *approach to query languages for relational databases.* The subsequent work of the author on Bounded Set Theory [33, 34, 36, 23, 22] is a natural continuation and generalisation of these results for the case of more flexible structures such as hereditarily-finite sets which are more suitable for providing mathematical foundations of *complex* or *nested databases.* Later these considerations absorbed, in [35], the idea of non-well-founded set (or *hyper-set*) theory introduced by P. Aczel [3]. This made the approach potentially applicable to (possibly distributed) *semistructured* or *Web-like databases* [24, 25, 37] with allowing cycles in hyper-links. Using *distributed, dynamically created agents* for more efficient querying of such databases by exploiting concurrently distributed computational resources (potentially over the whole Internet) is a further step to be developed within this framework.

Note, that our work *is not* intended to working out a *general theory* of agents. They are rather *a tool for achieving efficiency* in the querying process. However, an intermediate task consists of developing a new (or adapting some previously known) *theory or calculus of agents* suitable to our considerations. (Cf. references at the end of Section 4.) In this short paper we can only outline the main (essentially inseparable) ideas: *hyper-set approach to WDB* and to *querying WDB*, and *using agents* (in the context of this hyper-set framework). Also, in spite of any declared allusion to reality of WWW, which seems very useful and thought provoking, the present paper is highly abstract (and simultaneously informal and sketchy in some respects). But the author believes that the level of abstraction chosen is quite reasonable. The goal is not WWW itself, but some very general ideas around set-theoretic approach to WDB.

Our plan is as follows. Sect. 2 outlines the hyper-set approach to WDB. Sect. 3 is devoted to a brief but quite precise presentation of a hyper-set $\Delta$-language and its extensions for querying WDB. Sect. 4 discusses informally using agents for concurrent and distributed querying WDB. Then a fragment of a calculus of agents for computing $\Delta$-queries (a preliminary semiformal version) in Sect. 5 is presented. Finally, a Conclusion briefly summarises the whole paper.

## 2 An Outline of Hyper-Set Approach to WDB

The popular idea of a *semi- (or un-) structured data base* which, unlike a relational database, has no rigid schema, has arisen in the database community rather recently (cf. e.g. [1, 2, 7, 27, 8]), particularly in connection with the Internet. These databases can also be naturally described as *Web-like* databases (WDB) [24, 25, 37–39]. The best example of such a database is the World-Wide Web itself understood as a collection of Web-pages (`html`-files) distributed by various *sites* over the world and arbitrarily *hyper-linked*.

We adopt here a deliberately simplified, very abstract picture which, however, involves the *main feature* of WWW as based on hyper-links between URLs[1]

[1] Recall that in WWW URL = *Uniform Resource Locator*, i.e. *address* of a WWW-page (a specific html-file) on a Web-server.

of the Web-pages distributed potentially over the world. Having in mind set-theoretical approach to semistructured databases (arisen quite independently of WWW in [33] or, in somewhat different framework, [9, 10]) and contemporary ideas on semistructured databases (cf. op. cit.), we consider that *the visible part* of a WDB-page, Page($u$), with the URL $u$ is a (multi)set $\{\!|l_1, l_2, \ldots, l_k|\!\}$ of words $l_i$ (rather than a *sequence or list* $\langle l_1, l_2, \ldots, l_k \rangle$) where $u \stackrel{l_i}{\to} v_i$ represent all the outgoing hyper-links from $u$ to $v_i$. The URLs $v_i$ are considered as *non-visible* part of the *page* $\mathrm{Page}(u) = \{l_1 : v_1, l_2 : v_2, \ldots, l_k : v_k\}$ (a set of pairs of the kind *label* : *URL* — an abstraction from an `html` file). Both $l_i$ and corresponding $v_i$ are present in this page having the URL $u$, but in the *main window* of a browser we do *not* see the URLs $v_i$; $u$ being shown *outside* the window. Their role is very important for organisation of the information in WDB via *addressing/hyper-linking* but quite *different* from that of the words $l_i$, the latter carrying the *proper information.* All words $l_i$ on a Web-page are considered as (names or labels of) hyper-links, possibly trivial ones (e.g. links to an empty or non-existing page). By analogy with WWW we could underline $\underline{l_i}$ if $u \stackrel{l_i}{\to} v_i$ and $v_i$ is "non-empty", i.e. there exists at least one hyper-link $v_i \stackrel{m}{\to} w$ outgoing from $v_i$ (and it therefore makes sense to "click" on $\underline{l_i}$ to see all these $m$).

Such Web-like databases are evidently represented as directed *graphs with labelled edges* (or *labelled transition systems*), the labels $l$ serving as the names of hyper-links or *references* $u \stackrel{l}{\to} v$ between graph vertices (URLs) $u$ and $v$. Equivalently, WDB may be defined a finite subset of $\mathsf{URL} \times \mathsf{Label} \times \mathsf{URL}$. As in any database, a query language and corresponding software querying system are needed to properly query and retrieve the required information from a WDB.

In principle, for intermediate theoretical goals, it makes sense to consider also simplified "label-free", "pure" framework with simple unlabelled hyper-links $u \to v$ and (corresponding oversimplified) WDB is just an arbitrary finite (directed) graph with Edges — URLs. Then WDB-page Page($u$) with a URL $u$ is just a set of URLs $\{u_1, \ldots, u_k\}$ such that $u \to v_i$ are *all* outgoing edges from $u$.

Let us assume additionally that each graph vertex (URL) $u$ refers both to a *site* $\mathrm{Site}(u) \in \mathsf{SITES}$ where the corresponding WDB-page is saved as an (`html`) file and to this WDB-page itself. Some URLs may have the same site: $\mathrm{Site}(u_1) = \mathrm{Site}(u_2)$. However it is not the most interesting case, it is quite possible when there exists only one site for the whole (in this case non-distributed) WDB.

In the *hyper-set-theoretic* approach adopted here each graph vertex $u$ (URL [24, 37], called also *object identity, Oid*, [1, 2, 7]) is considered as carrying some *information content* $(\!|u|\!)$ — the result of some abstraction from the concrete form of the graph. In principle, two Web-pages with different URLs $u_1$ and $u_2$ may have the same (*hereditarily*, under browsing starting from $u_1$ and $u_2$) visible content. This is written as $(\!|u_1|\!) = (\!|u_2|\!)$ or, equivalently, as $u_1 \sim u_2$ where $\sim$ is a *bisimulation equivalence relation* on graph vertices (which can be defined in graph-theoretic terms [3] independently of any hyper-set theory). Somewhat analogous considerations based on a bisimulation relation are developed in [7], but the **main idea of our approach** consists not only in respecting the bisimulation (or, actually, *informational equivalence*) relation, but in "considering" graph vertices *as abstract sets* $(\!|u|\!)$ within hyper-set theory [3]:

$$(\!|u|\!) = \{l : (\!|v|\!) \mid \text{WDB has a labelled hyper-link } u \xrightarrow{l} v\}. \tag{1}$$

or in the simplified, "pure, label-free" semantics:

$$(\!|u|\!) = \{(\!|v|\!) \mid \text{WDB has a hyper-link } u \to v\}. \tag{2}$$

Thus, $(\!|u|\!)$ is a (hyper-) set of all labelled elements $l : (\!|v|\!)$ (also hyper-sets, etc.) such that WDB has a hyper-link $u \xrightarrow{l} v$. This (actually uniquely satisfying (1)) set-theoretic *denotational semantics* of graph vertices has evidently a complete agreement with the meaning of the equality $(\!|u_1|\!) = (\!|u_2|\!)$ or bisimulation equivalence $u_1 \sim u_2$ briefly mentioned above. (Cf. details in [35, 24, 25, 37].)

Unlike the ordinary (Zermelo-Frenkel) set theory, cycles in the membership relation are here allowed. Hence, a simplest such "cycling" set is $\Omega = \{\Omega\}$ consisting exactly of itself. Such sets must be allowed because hyper-links (in a WDB or in WWW) may in general comprise arbitrary cycles. We have *crucial theoretical benefits* from this hyper-set framework by using well understood languages (like $\Delta$ formally presented in Sect. 3) and ideas. Hyper-sets arising from graph vertices via (1) are quite natural and pose no essential conceptual difficulty.

Returning to the graph view, *querying* of a WDB may be represented as

- starting on a local site $s$ (where the query $q(u)$ is evaluated) from an *initial* or *input URL $u$ of a page* (saved as an `html` file on any remote site Site$(u)$);
- *browsing* page-by-page via hyper-links (according to the query $q$);
- *searching* in pages arising in this process (according to the query $q$);
- *composing new (auxiliary — for temporary intermediate goals — or final) hyper-linked pages with their URLs* (on the basis of the previous steps) with one of these pages declared as *main answer* or *resulting (output or "title") page* for the query;
- all of this may result in *reorganising of the data* (at least locally on $s$ by auxiliary pages located and hyper-linked between themselves and externally).

Of course, the user could habitually do all this job by hands. However, it is more reasonable to have corresponding implemented *query language and system* which would be able to formally express and evaluate any query $q$. This essentially differs from the ordinary search engines such as Alta Vista or Lycos, etc. Here we can use the advantages of our set theoretic view and of the corresponding language $\Delta$ [34, 36, 24, 25], at least theoretically.

Again, the **key point of our approach** consists in co-existing and interplaying *two* natural and related *viewpoints* for such a query $q$: graph- and set-theoretic ones. The above described steps result in some transformation (essentially a local extension by new pages) of the original WDB (WWW) graph. Usually, in semistructured databases *arbitrary* graph transformations are allowed [1, 2]. But if this process *respects the information content* $(\!|u|\!)$ of the input WDB vertices (URLs) $u$ then it should be restricted to be *bisimulation invariant* (as it was done also independently in [7], but without a sufficient stress on set theory) and therefore it could be also considered as a *set theoretic operation* $q$, i.e.

$$q : (\!|\text{input URL}|\!) \longmapsto (\!|\text{output URL}|\!) = q(\!|\text{input URL}|\!).$$

Originally, this approach arose exactly as a set-theoretical view [33, 34] with (at that time *acyclic*) graphs representing the ordinary or *well-founded hereditarily-finite sets* whose universe is denoted as **HF** [2] [6]. Any database state may be considered as a set of data each element of which is also a further set of data, etc. (with always *finite depth of nesting* which is equal 4 in the case of a "flat" *relational database*). A generalised universe of *anti-founded hereditarily-finite (hyper) sets* is called **HFA**, and any query is considered as a map (set-theoretic operation — a well understood concept) $q : \mathbf{HF} \to \mathbf{HF}$ or $q : \mathbf{HFA} \to \mathbf{HFA}$.

A (version of) purely set-theoretic *query language* $\Delta$ allowing expression of queries $q$ to a WDB has been developed (with a natural and simplest syntax whose main feature is the use of *bounded* quantifiers $\forall x \in t$ and $\exists x \in t$ and other, essentially bounded, constructs; cf. e.g. [35, 36, 25, 24] and Sect. 3 below for the details). This language has *two kinds of semantics* in terms of: (i) *set-theoretic operations* $q$ over **HFA** (such us set union or "concatenation" of WDB-pages), etc. — *a high level* semantics *for humans*) and (ii) corresponding *graph (WDB) transformers* $Q$ — (*a lower level* semantics *for program implementation*), with a commutative diagram (3) witnessing that both semantics agree

$$\begin{array}{ccc} \mathbf{HFA} & \xrightarrow{q} & \mathbf{HFA} \\ {\scriptstyle (\!|-|\!)}\uparrow & & \uparrow{\scriptstyle (\!|-|\!)} \\ \mathcal{WDB} & \xrightarrow{Q} & \mathcal{WDB} \end{array} \qquad q(\!|\mathrm{WDB}|\!) = (\!|Q(\mathrm{WDB})|\!) \tag{3}$$

and $Q$ is *bisimulation invariant* or *respecting informational equivalence*. Here $\mathcal{WDB}$ is the class of all WDBs (i.e. finite labelled or unlabelled graphs) with a distinguished URL (vertex) $u$ in each, to which $(\!|-|\!)$ is actually applied.

The *expressive power* of this language and its versions was completely characterised in terms of PRIME- (both for **HF** and **HFA**) and (N/D)LOGSPACE- (for **HF**) computability over graphs via (3) [33, 34, 24, 25, 23, 22]. It is because of flexibility, naturalness and reasonable level of abstractness of our set-theoretic approach to WDB (which seemingly nobody else used in the full power) such expressibility results where possible. Here it is probably suitable to mention, for the contrast, a quotation from [8], p. 14: "It should be noted that basic questions of expressive power for semistructured database query languages are still open".

[2] **HF** and **HFA** below may be considered in *two versions*: (i) *labelled* one when a set in the universe consists of labelled elements (which can be also such sets, or, may be, *urelements* or *atoms*), and (ii) *pure, atoms and labels-free* version — the ordinary pure sets of sets of sets, etc., the empty set being the simplest object. For simplicity, $\Delta$-language and calculus of agents presented in Sect. 3 and 5, respectively, are devoted to the pure case. Labelled case would seemingly add nothing essential to our agents approach. Anyway, it is better to start with the simplest case.

A preliminary *experimental implementation* of a version of $\Delta$ [38] has been developed (when the author worked) in the Program Systems Institute of Russian Academy of Sciences (in Pereslavl-Zalessky) as a query language for the "real" WWW which is based on the steps of browsing, searching, etc. described above.

## 3 Hyper-set $\boldsymbol{\Delta}$-Language for Querying WDB

### 3.1 Basic $\boldsymbol{\Delta}$-Language for the Pure HF or HFA [3]

**Definition 3.1.** Define inductively $\Delta^*$*-formulas* and $\Delta^*$*-terms* by the clauses

$$\langle \Delta^*\text{-terms}\rangle ::= \langle\text{variables}\rangle \mid \emptyset \mid \{a,b\} \mid \bigcup a \mid \{t(x) \mid x \in^{(*)} a \mathbin{\&} \varphi(x)\}$$

$$\langle \Delta^*\text{-formulas}\rangle ::= a = b \mid a \in^{(*)} b \mid \varphi \mathbin{\&} \psi \mid \neg\varphi \mid \forall x \in^{(*)} a \varphi(x)$$

where $\varphi$ and $\psi$ are any $\Delta^*$-formulas, $a, b$ and $t$ are any $\Delta^*$-terms and $x$ is a variable not free in $a$. The parentheses around $*$ in $\in^{(*)}$ mean that there are two versions of the membership relation: $\in$ and its non-reflexive *transitive closure* $\in^*$. We write $\Delta$ when $\in^*$ is not used at all. The sub-language $\Delta$ corresponds to the *basic* [17] or *rudimentary* [21] set-theoretic operations.

| |
|---|
| From the database's point of view, (labelled version of) the language $\Delta$ is analogous to *relational calculus*, however it is devoted to *arbitrary sets of sets of sets*, etc. rather than to relations — also *sets of a specific kind.* |

The main idea is that $\Delta$-formulas involve only *bounded* quantification $\forall x \in^{(*)} a$ and $\exists x \in^{(*)} a \rightleftharpoons \neg\forall x \in^{(*)} a\neg$ with the evident meaning. All other constructs of $\Delta$ are also in a sense "bounded". But, for example, the unrestricted *power-set operation* $\mathcal{P}(x) \rightleftharpoons \{y : y \subseteq x\}$ is considered as intuitively "unbounded", "impredicative", computationally intractable, and therefore it is not intended to be taken as *natural* extension of $\Delta$.

For simplicity, as common, we shall use set variables, $\Delta^*$-terms and formulas *both as syntactic objects and as denotations of their values* in **HF** (**HFA**). For example, $\Delta$-separation $\{x \in a : \varphi(x)\}$ for $\varphi \in \Delta$ gives 'the set of all $x$ in the set $a$ for which $\varphi(x)$ holds' and is a special case of the construct $\{t(x) : x \in a \mathbin{\&} \varphi(x)\}$ = 'the set of all values of $t(x)$ such that …'. Also $x \in \{a, b\}$ iff $x = a$ or $x = b$, $x \in \bigcup a$ iff $\exists z \in a(x \in z)$ and $\in^*$ is a transitive closure of the membership relation $\in$ on **HF** (**HFA**), i.e. $x \in^* y$ iff $x \in x_1 \in x_2 \in \ldots \in x_n \in y$ for some $n \geq 0$ and $x_1, \ldots, x_n$ in **HF** (**HFA**). In particular, define in $\Delta^*$ the *transitive closure of a set* $y$ as $\mathrm{TC}(y) \rightleftharpoons \{y\} \cup \{x : x \in^* y\}$.

Any $\Delta$-term $t(\bar{x})$ evidently defines a set-theoretic operation $\lambda\bar{x}.t(\bar{x}) : \mathbf{HFA}^n \to \mathbf{HFA}$. For example, let $u \cup v \rightleftharpoons \bigcup\{u, v\}$, $u \cap v \rightleftharpoons \{x \in u : x \in v\}$, $u \setminus v \rightleftharpoons \{x \in u : x \notin v\}$, $\{u\} \rightleftharpoons \{u, u\}$, $\{u_1, u_2, \ldots, u_k\} \rightleftharpoons \{u_1\} \cup \{u_2\} \cup \ldots \cup \{u_k\}$. Ordered pairs and related concepts are defined in $\Delta$ as $\langle u, v\rangle \rightleftharpoons \{\{u\}, \{u, v\}\}$, or (in the labelled case), more naturally from a practical point of view, as $\{\mathrm{Fst} : u, \mathrm{Snd} : v\}$ for some two special labels (attributes) Fst and Snd, etc.

[3] See [34] for generalisation of $\boldsymbol{\Delta}$ for the case of **HF** and **HFA** with urelements (atoms) and labels (attributes).

### 3.2 Extensions of $\Delta^*$

There are important extensions of $\Delta^*$-language corresponding to PTIME or (N/D)LOGSPACE computability over **HFA**, resp., **HF**, as we mentioned in Sect. 2. Due to lack of space, we will consider only those giving rise to PTIME.

**Decoration** (Aczel, [3])

$$\mathbf{D} : \mathbf{HFA} \to \mathbf{HFA}, \quad \mathbf{D}(\langle g, v\rangle) = (\!| v |\!)_g$$

with $g \in \mathbf{HFA}$ any graph (set of ordered pairs in set-theoretic sense). The decoration operation $\mathbf{D}$ embodies Aczel's **Anti-Foundation Axiom**.

**Collapsing** (Mostowski, [6])

$$\mathbf{C} : \mathbf{HF} \to \mathbf{HF}$$

is partial case of $\mathbf{D} : \mathbf{HFA} \to \mathbf{HFA}$ when $g$ is any *well-founded*, i.e. *acyclic* graph (in the finite case considered).

**Recursive $\Delta$-Separation** term construct **Rec**:

$$\textbf{the-least}\ p.[p = \{x \in a : \varphi(x, p)\}]$$

for $\varphi \in \Delta$ *positive* (and with an additional requirement) on set variable $p$.

## 4 Using Agents for Concurrent Querying WDB

**What is new in the present approach.** The *main problem* with implementing the set-theoretic language $\Delta$ is that (potentially) a *significant amount of browsing* may be required during query evaluation. In the real Internet the result of each mouse click on a hyper-link (downloading a page from remote site) is delayed by several seconds at best and the whole time of querying may take hours (despite all other pure computation steps being much faster). Therefore, some *radical innovation in the implementation is necessary* to overcome the problem of multiple browsing via the Internet (in a *non-efficient, sequential* manner).

A well-known technique, used by most search engines such as Alta Vista, is that of creating (and permanently renewing) an *index file* of the WWW to which queries are addressed and some parts of which may be actually rather old.

The approach described here is aimed at "goal-directed" querying the *real* Web, *as it is* at the current moment, where *reorganising the data* (not only searching) is an additional and crucial feature of the query language $\Delta$.

As a reasonable solution, we suggest using **Dynamic Agent Creation**. The set-theoretic language $\Delta$ appears very appropriate for this goal.

**Agents as $\Delta$-terms.** Each agent, i.e. essentially a $\Delta$-term (= a query $q = q(u)$ written in $\Delta$-language) having an additional feature of an *active behaviour*, when starting querying from "his" local site $s$, may also *send* (via the Internet) to remote sites $s_1, s_2, \ldots$ some appropriate sub-terms $p_1, p_2, \ldots$ which, as *agents*, will work analogously on these remote sites, i.e., if necessary, will send new *descendant agents*, etc. Eventually, the main agent will collect all the data obtained from this process. Potentially, the *whole Internet* would *concurrently participate* in the *distributed* evaluation of the given query. We expect that this will essentially *accelerate querying*. One of medium-term goals is to establish the truth of this expectation in a theoretical framework. Another goal is producing an experimental agent based implementation of the language $\Delta$ to check practically this expectation and to get more experience for further developing this approach.

**A Typical Example.** To calculate the set-theoretic $\Delta$-term (in unlabelled case)

$$q = \{p(v) \mid v \in a\},$$

i.e. the "set of all $p(v)$ such that $v$ is in $a$", this term, as an "agent", *sends many descendant agents/sub-terms* $p(v)$ for all (URLs) $v$ contained on the page $a$ to the various, probably *remote*, sites $\mathrm{Site}(v)$ of the WWW to which all these (URLs) $v$ refer. When each agent $p(v)$ finishes "his" computation (possibly with the help of their descendant agents), "he" will send the result to the "main agent" $q$, "who" will appropriately collect all the results together as a new (URL — the value of $q$ — and corresponding `html` file of a) Web-page containing all the computed URLs $p(v)$ for all $v \in a$. Each agent $q$, $p(v)$, for all $v \in a$, etc. will resolve "his" own task, which may be probably not so difficult and burdensome for the site resources where the agent works, since "his" descendant agents will do the rest of the job possibly on other sites. If, by some reason, the "permission is denied" to send a descendant agent to a remote site, the parent agent will also take corresponding part of the job and will do it "himself" from "his" local site — by browsing, searching, etc.

**The further and most crucial goal** of the described work therefore consists in *formalising the ideas above*. It may be done in the form of a specially elaborated (or suitably adapted for the query language $\Delta$ considered here) *calculus of agents* which describes as descendant agents are creating, moving around, doing their job, communicating their result to the "senior" agents by which they were created, until the result of the initial query is obtained. In particular, this will give *agent based graph transformer operational semantics* for the language $\Delta$. Some appropriate version of the corresponding agent calculus (for a partial case covering the above example) will be presented in Sect. 5. We realize that this is probably not the unique possible version. Cf. discussion below.

*Correctness* of this semantics with respect to natural (hyper) set-theoretical semantics must be proved. It should be shown rigorously that the *time complexity* of this approach with agents is really better than that without agents (i.e. essentially with only one main agent which creates no sub-agents and makes all "his" job of browsing, searching and creating pages alone). In particular, it is necessary to formulate and develop a reasonable approach to time and space complexity for agent based querying (cf. the next paragraph) and then to estimate and compare time and space complexity of queries either when agents are used or not. In the ideal, the resulting agent calculus should be *capable of being implemented* as a working *query system* to a WDB, say, to the WWW.

Note that an appropriate abstraction from querying of the WWW should be found. For example, in reality, each step of browsing requires some physical, actually unpredictable time. For simplicity it may be taken that each step of browsing or exchanging information between agents takes *one unit of time* and all other computation steps (of searching some information trough Web-pages and composing new ones) cost nothing (in comparison with the browsing). Also when several agents are sent to the same Web-site for their work their activity may be considered either as sequential or in an interleaving manner according to communication with other agents. Space complexity also may be considered in a simplified version, e.g. as the maximum number of agents working simultaneously. The simultaneity concept should be also defined (approximated) appropriately as the run of time in the model may not correspond to the real one, as we noted above. On the other hand, how much of memory resource each agent needs on its site (server) could additionally be taken into account. All of this seems has no canonical or simple, most elegant theoretical solution.

**Some related work** (i) on *communication and concurrency* by R. Milner and P. Aczel, [28, 4, 5], (ii) on *mobile agents (ambients)* by R. Milner et al. [29], L. Cardelli [8], C. Fournet et al. [16], (iii) on *logical specification* of agents' behaviour by M. Fisher and M. Wooldridge [15], (iv) on *distributive querying* by D. Suciu [40, 41] and A. Sahuguet et al. [31], (v) on corresponding application to agents of Y. Gurevich's *Abstract State Machines* [13], (vi) on *Mobile Maude* by J. Meseguer et al. [12], (vii) on *Nomadic Pict* [30], (viii) on a project *LOGIC Programming for the World-Wide WEB* by A. Davison et al. [11] and probably many others could be useful in our specific set-theoretic context either ideologically, or by direct using corresponding formalisms or their analogues.

However, in our very concrete situation it seems reasonable to present a calculus of agents from the principles which the query language $\Delta$ almost immediately "dictates" according to its natural and clear semantics, postponing for a while comparison with other formalisms and ideas, because here we will present only rather preliminary, even not completely rigorous version of a calculus.

**Further perspectives from the practical point of view.** It is clear, that *the whole approach depends on giving permissions* by various Web sites for agents to work there. For a WDB on a local net or on an *Intra*net this problem can be resolved by an agreement with the administrator of the net. However, for the case of the global Internet the whole Internet community should agree on a *general standard* for such permissions with an appropriate guarantee of no danger from the "arriving" agents. Of course this may depend on whether this framework will be sufficiently successful for local nets, as well as on the current state of affairs concerning programming of distributed computation in the Internet.

## 5 A Fragment of a Calculus of Agents for Computing $\Delta$-Queries (Preliminary Semiformal Version)

According to our informal description in the previous section, each Web-site $s$ (server, or even any computer with a Web-browser)[4] may contain agents/queries $q$??, or $q$ = ?? working on the site $s$ (where $q$, with some exceptions, is a $\Delta$-term or $\Delta$-formula; the general form of an agent/query is presented in (4)). All queries on a site $s$ constitute a *queue* $\tilde{q}$ according to which (from left to right) they should be evaluated. We consider the expression $s\,\tilde{q}$ as a current *state* of a site $s$ w.r.t. (the state of) its queries in $\tilde{q}$.

The *double question mark* "??" means that the query/agent is waiting for the server $s$ to start (or allow) executing it, while a *single question mark* "?" corresponds to an agent $q$ which has already started to work, but at present is *awaiting results* from its subqueries (descendant agents) which have been sent to other sites. Also, we use single "?" for so-called "passive" queries which are only waiting for results from "active queries" (cf. below).

Thus, the sequence $\tilde{q}$ consists of *active agents/queries* (syntactically more complicated than just "$q$ = ??") formally written in round brackets as:

$$(q_i = \epsilon_i \to s_i\, id_i), \quad i = 1, \dots, m. \tag{4}$$

Here "$\to s_i\, id_i$" is read as "send the result of the query to the site $s_i$ by the address $id_i$". Expressions like $id$ of our agents metalanguage are *query identities* of various "passive" subqueries (to be considered below — they are parts of $\epsilon$-expressions of other, actually parent agents/queries also of the form (4)). These identities are needed to distinguish *occurrences* of "passive" subqueries and also to serve as their *addresses* in the corresponding queues (the current implicit queue in the site $s_i$, in the case of (4)). All query identities constitute a set $\mathsf{Id}$ of finite strings consisting of some standard computer keyboard symbols starting with, say, # and containing no more #s.

Some special identities written as #`browser-of-` < `user` > are reserved as *special, exceptional cases*, used for queries of the form

$$(q_i = ?? \to s'\#\texttt{browser-of-} < \texttt{user} >)$$

in the site-queue $s\,\tilde{q}$ initiated by some *(human) user/agent* working on another site $s'$ (his PC). This query will be started for execution on the site $s$ (according to our formal calculus of agents to be partially described below). Let us assume that some URL $u_i$ will be obtained by evaluating (probably in many steps) the query "$q_i$ = ??", giving rise to the result "$q_i = u_i$". Then, the remaining part "$\to s'$#`browser-of-` < `user` >" means that the resulting Page($u_i$) should be opened in a new window of a browser on the site $s'$ with some additional special subwindow in the browser where the answer "`QUERY RESULT OF <user>`

[4] Everything is still considered in a very abstract way; our Web or Internet terminology is used mainly to provide a general intuition rather than a complete or very precise correspondence to standard concepts.

`IS:` $q_i = u_i$" to the query is explicitly written for the user's convenience to understand why this (perhaps suddenly appearing because of a possible time delay) browser window and its content Page$(u_i)$ arose.

Let us *define in general* what the expression $\epsilon_i$ in (4) may be:

- double question mark "??", or
- just some *resulting* $u_i \in \mathsf{URL}$, if $q_i$ is $\Delta$-term (a normal query), or a *resulting* boolean value $\mathsf{true}$ or $\mathsf{false}$, if $q_i$ is $\Delta$-formula (a boolean query), or
- starting with a single question mark "?" and possibly with a URL, a *sequence* of auxiliary *passive queries* or corresponding *answers* in square brackets

$$[id_r : r = ?] \quad \text{or} \quad [id_r : r = u_r], \quad \text{or} \quad [id_r : r = U_r], \tag{5}$$

  with $u_r \in \mathsf{URL}$, $U_r$ a finite set of URLs or some simple expressions consisting of URLs (in the last two cases, the "$id$ :" will usually be omitted); these passive queries or answers are needed to calculate the answer to the main query (4) which is of the form

$$(q_i = ?[\ldots]\ldots[\ldots] \to s_i\, id_i) \quad \text{or} \quad (q_i = ?url[\ldots]\ldots[\ldots] \to s_i\, id_i); \tag{6}$$

  moreover, all query identities $id_r$ should be *different* in passive queries of the whole queue $\tilde{q}$ (to serve as unique addresses of these passive queries for sending the answers from other sites).

Here a single "?" at the beginning of $\epsilon_i$ indicates that the result of $q_i$ in (4) or (6) is still unknown. Even if we have some $url$ in (6), the content of the Page$(url)$ may be not completed yet. To get the result of $q_i$ in (6) some further actions of this agent may be required even if all passive subqueries are already answered and replaced by the results of the form $[r = u_r]$ or $[r = U_r]$.

As the server $s$ may be "busy" with its own work or evaluating queries from the queue $\tilde{q}$, it is indeterminate when it will execute the first "waiting" query $q_i$?? or $q_i$? (with queries for all $j < i$ having already some final answers $u_j \in \mathsf{URL}$ or a boolean value $\mathsf{true}$ or $\mathsf{false}$, for boolean queries, or waiting the results from their descendant subagents which already have been sent to some sites by the agent $q_j$ with the help of the server $s$). Therefore we should have nondeterministic, asynchronous and concurrently applied *reduction rules* for the extended sites $s\tilde{q}$.

Note that all $\Delta$-queries $q_i$ should be closed terms or formulas with free variables substituted by some $url$ s (the addresses of WDB-pages which simultaneously denote corresponding sets $(\!|url|\!)$ according to (1) or (2) — in our present "pure" case). However, $q_i$ may be also a specially created agent which is not formally a $\Delta$-term.

Some $q_i$ may be just a constant $u \in \mathsf{URL}$. This query does not require evaluation. As a URL, it is the final result of evaluation of the query. Therefore let us introduce the **trivial rule** for such *atomic* $q_i$:

$$s\tilde{q}(u = ?? \to s'id')\tilde{q}' \longmapsto s\tilde{q}(u = u \to s'id')\tilde{q}'. \tag{7}$$

The following **double structural rule** (*inseparable* into two independent parts; same for other "multiple rules" below presented in curly brackets), for the sequential replacement of queues in two sites, shows what happens after a query agent $q_i$ has received an answer $u_i$ (say, as in the trivial case of (7)):

$$\left\{\begin{array}{ll} s\tilde{q}(q_i = u_i \to s_i id_i)\tilde{q}' \longmapsto & s\tilde{q}\tilde{q}', \\ s_i\tilde{q}'' & \longmapsto s_i\langle u_i \to id_i\rangle\tilde{q}'' \end{array}\right\} \tag{8}$$

(if $s = s_i$ then it is naturally assumed that $\tilde{q}'' = \tilde{q}\tilde{q}'$ should hold) where the *substitution operator* $\langle u_i \to id_i\rangle$ means that the (unique) subquery $[id_i : \ldots = ?]$ (of some parent agent) with the same $id_i$ should be found in an $\epsilon$ part of $(q = \epsilon \to \ldots)$ occurring in $\tilde{q}''$ and corresponding "?" should be replaced by $u_i$ with $id_i$ possibly omitted as already used and unnecessary anymore, giving result $[\ldots = u_i]$ in $\epsilon$ in the place of the passive query $[id_i : \ldots = ?]$ occurrence.

### 5.1 Rules for the Query "$\{p(v) \mid v \in a\} =??$"

The following three rules are devoted to the query "$\{p(v) \mid v \in a\}??$", informally considered in the previous section.

**The first rule** postpones the main query in favour of $(a = ?? \to s\, id_a)$ sent to the same site $s$, last in the queue:

$$\begin{array}{l} s\tilde{q}(\{p(v) \mid v \in a\} = ?? \to \ldots)\tilde{q}' \longmapsto \\ \qquad s\tilde{q}(\{p(v) \mid v \in a\} = ?[id_a : a = ?] \to \ldots)\tilde{q}'(a = ?? \to s\, id_a) \end{array}$$

As discussed above, the part "$\to s\, id_a$" shows where the result should be sent; here, to the same site and queue to the passive query (in the main query) labelled by $id_a$. Note, that expressions $[id_a : a = ?]$ and $(a = ?? \to s\, id_a)$ play different roles in the queue (passive and active, respectively). When the latter will be resolved, with some result $u_a \in \mathsf{URL}$, it can be annihilated, as in (8), after the "?" in the former expression will be replaced by the result $u_a$ obtained.

**The second, multiple rule** has additionally a *premise condition*[5] assuming that on the site $s$ the query/agent $(\{p(v) \mid v \in a\} = ?[a = u_a] \to \ldots)$ (already "knowing" that $u_a$ is the value of $a$) first inspects the content of the $\mathrm{Page}(u_a)$ (say, with the help of a browser on the site $s$). Then, depending on the result in the premise, the following multiple rules under the horizontal line are performed:

$$\frac{\mathrm{Page}(u_a) = \{u_1, \ldots, u_k\}}{\left\{\begin{array}{l} s\tilde{q}(\{p(v) \mid v \in a\} = ?[a = u_a] \to \ldots)\tilde{q}' \longmapsto \\ s\tilde{q}(\{p(v) \mid v \in a\} = ?[a = u_a][id_1 : p(u_1) = ?] \ldots [id_k : p(u_k) = ?] \to \ldots)\tilde{q}' \\ \\ \qquad \mathrm{Site}(u_i)\tilde{q}_i \longmapsto \mathrm{Site}(u_i)\tilde{q}_i(p(u_i) = ?? \to s\, id_i), \quad i = 1, \ldots, k. \end{array}\right\}}$$

Again, the main query is postponed until new descendant agents of the form $(p(u_i) = ?? \to s\, id_i)$ (depending on the result $u_a$ of the query "$a = ??$", or, more precisely, on the content of $\mathrm{Page}(u_a)$) are created and sent to the sites $\mathrm{Site}(u_i)$ (with their current queues $\tilde{q}_i$) to work there and to calculate values of $p(u_i)$.

[5] We could use instead an if-then construct or just composition of some actions.

**Finally**, the following rule allows the complete evaluation of the query "$\{p(v) \mid v \in a\} = ??$":

$$\frac{\mathrm{Page}(u_a) = \{u_1, \dots, u_k\}, \quad \texttt{new}\ \mathrm{Page}(u) = \{u'_1, \dots, u'_k\}}{\begin{array}{c} s\tilde{q}(\{p(v) \mid v \in a\} = ?[a = u_a][p(u_1) = u'_1] \dots [p(u_k) = u'_k] \to \dots)\tilde{q}' \longmapsto \\ s\tilde{q}(\{p(v) \mid v \in a\} = u \to \dots)\tilde{q}' \end{array}}$$

where $u$ is a *new* URL on the site $s$ ($\mathrm{Site}(u) = s$) such that $\mathrm{Page}(u) = \{u'_1, \dots, u'_k\}$. It is assumed that the current agent, with all passive queries answered,

$$(\{p(v) \mid v \in a\} = ?[a = u_a][p(u_1) = u'_1] \dots [p(u_k) = u'_k] \to \dots),$$

will create this new page on its site $s$ which will automatically receive a URL $u$ (depending on the directory where corresponding `html` file with this page will be saved and, of course, on the name of this file — as usual for URLs).

### 5.2 Rules for the Transitive Closure TC

The rules for calculating $\mathrm{TC}(a)$ are based on the following simple ideas. Starting from the calculation of URL $u_a = a$, numerous identical copies of a special agent $\mathrm{TC}'$ are sent step-by-step (first by TC, and then by descendant $\mathrm{TC}'$s) to (the sites of) descendants $v$ of the vertex $u_a$ in the given graph of the WDB. Each copy $\mathrm{TC}'(v)$ gathers URLs immediately accessible from $v$ and sends them, with the information how they were obtained, to the main agent $\mathrm{TC}(a)$ which collects all of this information in a newly created file $\mathrm{Page}(u)$ with a URL $u$. When all URLs reachable from $u_a$ are collected in $\mathrm{Page}(u)$, this parallel process is considered finished, and gives the result $u$.

**The first rule** is as in the previous subsection:

$$\begin{array}{l} s\tilde{q}(\mathrm{TC}(a) = ?? \to \dots)\tilde{q}' \longmapsto \\ \qquad s\tilde{q}(\mathrm{TC}(a) = ?[id_a : a = ?] \to \dots)\tilde{q}'(a = ?? \to s\, id_a) \end{array}$$

**The second, multiple rule** has an additional *premise condition*:

$$\frac{\mathrm{Page}(u_a) = \{u_1, \dots, u_k\} \quad \texttt{new}\ \mathrm{Page}(u) = \{u_a \to u_1, \dots, u_a \to u_k\}}{\left\{ \begin{array}{l} s\tilde{q}(\mathrm{TC}(a) = ?[a = u_a] \to \dots)\tilde{q}' \longmapsto \\ \qquad\qquad s\tilde{q}(\mathrm{TC}(a) = ?u[a = u_a][id_{\mathrm{TC}} : \mathrm{TC}'(\ ) = ?] \to \dots)\tilde{q}' \\ \\ \mathrm{Site}(u_i)\tilde{q}_i \longmapsto \mathrm{Site}(u_i)\tilde{q}_i(\mathrm{TC}'(u_i) = ?? \to s\, id_{\mathrm{TC}}), \quad i = 1, \dots, k. \end{array} \right\}}$$

Again, the main query is postponed until new descendant agents of the form $(\mathrm{TC}'(u_i) = ?? \to s\, id_i)$ (depending on the result $u_a$ of the query "$a = ??$") are sent to the sites $\mathrm{Site}(u_i)$ (with their current queues $\tilde{q}_i$) to work there and to calculate values of $\mathrm{TC}'(u_i)$. Here $\mathrm{TC}'(\ )$ is a new agent generated by TC. URLs $u_i$ serve as *parameters* for $\mathrm{TC}'$. We also use $\mathrm{TC}'(\ )$ *without parameters* in the corresponding passive queries. Further, $u$ is a *new* URL on the site $s$ ($\mathrm{Site}(u) = s$) such that (at the beginning of calculating of $\mathrm{TC}(a)$) $\mathrm{Page}(u) = \{u_a \to \mathrm{Page}(u_a)\} = \{u_a \to u_1, \dots, u_a \to u_k\}$ (a set of arrows between URLs).

We see that the format of $\epsilon$ expressions is slightly extended in comparison with the previous subsection. However here we seemingly have an answer $u$ to the question "$\mathrm{TC}(a) = ??$", the question mark before $u$ indicates that *this answer is only a partial one.* More precisely, it is partial only in the content of the corresponding $\mathrm{Page}(u)$ ( $= \{u_a \to u_1, \ldots, u_a \to u_k\}$ at the first moment). It should be extended with the help of descendant agents of the form $\mathrm{TC}'(v)$.

Moreover, we need to consider, *more generally,* that the answer to any subquery $[\mathrm{TC}'(v) = ?]$ (first, $v = u_i$ from the above rule) is the set of arrows between URLs $U = \{v \to \mathrm{Page}(v)\} = \{v \to v_1, \ldots, v \to v_n\}$, where $\mathrm{Page}(v) = \{v_1, \ldots, v_n\}$ (we let $U = \{v \to \}$ if $\mathrm{Page}(v) = \emptyset$, i.e. $v$ has no children) whereas in the previous subsection it was assumed that only some URLs may be query answers.

**The third, also multiple rule** describes the behaviour of the agent $\mathrm{TC}'$ which (i) creates a set of arrows $v \to v_i$ outgoing from the URL $v$ under consideration (as described above), and (ii) propagates its copy to sites $\mathrm{Site}(v_i)$. From each site where (a copy of) $\mathrm{TC}'$ is working, it sends the results to the same place $s\,id_{\mathrm{TC}}$.

$$\frac{\mathrm{Page}(v) = \{v_1, \ldots, v_n\} \quad U = \{v \to \{v_1, \ldots, v_n\}\}}{\left\{\begin{array}{l} \mathrm{Site}(v)\,\tilde{q}(\mathrm{TC}'(v) = ?? \to s\,id_{\mathrm{TC}})\tilde{q}' \longmapsto \\ \qquad\qquad\qquad\qquad \mathrm{Site}(v)\,\tilde{q}(\mathrm{TC}'(v) = U \to s\,id_{\mathrm{TC}})\tilde{q}' \\ \\ \mathrm{Site}(v_i)\tilde{q}_i \longmapsto \mathrm{Site}(v_i)\tilde{q}_i(\mathrm{TC}'(v_i) = ?? \to s\,id_{\mathrm{TC}}), \quad i = 1, \ldots, n. \end{array}\right\}}$$

**The rule continuing partial evaluation** of the query "$\mathrm{TC}(a) = ??$" (extending the content of $\mathrm{Page}(u)$):

$$\frac{\mathrm{Page}(u) := \mathrm{Page}(u) \cup U}{\begin{array}{r} s\,\tilde{q}(\mathrm{TC}(a) = ?u[a = u_a][id_{\mathrm{TC}} : \mathrm{TC}'(\,) = U] \to \ldots)\tilde{q}' \longmapsto \\ s\tilde{q}(\mathrm{TC}(a) = ?u[a = u_a][id_{\mathrm{TC}} : \mathrm{TC}'(\,) = ?] \to \ldots)\tilde{q}' \end{array}}.$$

Here $u$ is (intended to be) the URL created above on the site $s$ ($\mathrm{Site}(u) = s$) such that $\mathrm{Page}(u)$ is the union of the old version and the set of arrows, say, $U = U_i = \{u_i \to \mathrm{Page}(u_i)\}$. The passive query $[id_{\mathrm{TC}} : \mathrm{TC}'(\,) = ?]$ appears again to wait some new $U$ from other $\mathrm{TC}'(v)$.

Finally, we present a **rule to complete the calculation of** $\mathrm{TC}(a)$:

$$\frac{\left\{\begin{array}{c} \text{Assuming all participating URLs in } \mathrm{Page}(u) \text{ occur as the left ends of arrows,} \\ \textbf{the optional action: } \text{omit in this page all arrows} \\ \text{and duplication of all remaining URLs} \end{array}\right\}}{s\,\tilde{q}(\mathrm{TC}(a) = ?u[a = u_a][id_{\mathrm{TC}} : \mathrm{TC}'(\,) = ?] \to \ldots)\tilde{q}' \longmapsto s\,\tilde{q}(\mathrm{TC}(a) = u \to \ldots)\tilde{q}'}.$$

The premise condition guarantees that $\mathrm{Page}(u)$ is completed as required, giving the whole transitive closure of $a$ (or $u_a$), possibly with some arrows making $\mathrm{Page}(u)$ a graph. In the latter case, given any $a$ and $b$, we could use the corresponding versions of $\mathrm{TC}(a)$ and $\mathrm{TC}(b)$ to calculate the bisimulation relation between corresponding graphs and then, if needed, to evaluate also queries "$(a = b)??$" and "$(a \in b)??$".

We postpone considering rules, their correctness and conditions and possible exceptions of applicability for the above and other constructs of the $\Delta$-language until a more advanced version of this paper will be prepared (with a bigger volume allowed). Even in this form they demonstrate how the mobile and propagating agents can work over a (potentially World-Wide) distributed Web-like hyper-set database in a *concurrent, asynchronous and distributed manner* to evaluate set-theoretically described queries. After developing a full version of the calculus of "$\Delta$-agents" both *theoretical* and *practical* (directed towards implementation) investigations can be carried out independently.

### 5.3 What Makes Agents Active, or How Could they be Implemented?

Of course, agents, as they are described above, are only syntactic expressions. To make them active and really mobile, concurrent and working distributively, we need a unique program *Distributed Query Evaluator* DQE installed (desirably) *on each* WDB (or WWW) site which will fulfil all the necessary rules of manipulating queues (and agents in them). Of course, it should be sufficiently flexible to work even if some sites do not have installed DQE or do not currently allow our agents. Also it should be able to redirect some agents to other neighbour or "friendly" sites if the queue on the current site is too long, etc. Only in this case will the whole enterprise of making querying in $\Delta$ *considerably more efficient* according to the ideas presented in this paper be successful. It is really a difficult problem to make such a program which the whole Internet community and computer and software companies will adopt like well-known browsers (Internet Explorer or Netscape). Anyway, any new step in such a direction (theoretical, experimental, practical) **to make the Internet a big community of computers helping one another** seems a reasonable (much more general than our present) task.

## 6 Conclusion

The starting point for this research was a mostly theoretical one related to descriptive complexity theory and to extending its methods to relational, nested, complex and Web-like distributed databases grounded on (hyper) set theory. Our present work is also theoretical, but it is directed towards the problem of developing a necessary radical step for more efficient implementation. Considering dynamically created agents *in full generality*, working and communicating concurrently, asynchronously and distributively over the Internet *is not* the immediate goal of research here. We rather represent a *machinery for achieving the efficiency of querying in the present (hyper)set approach*. As a result, this work may be of direct benefit to research communities on semi-structured and distributed databases and on multi-agent systems. It also has the potential of longer term benefit to the Internet community. More concretely, it may lead to better understanding of the basic principles of efficient querying of WDB and to an experimental prototype of such a query system.

## 7 Acknowledgements

The author is very grateful to Michael Fisher, Alexander Bolotov, Chris Brand and Grant Malcolm for fruitful discussions on the approach presented and for the help in polishing English of of this paper. He is especially thankful to his colleagues from Pereslavl-Zalessky (Russia) Alexander Leontjev, Alexei Lisitsa and Yuri Serdyuk actively participating in set-theoretic approach to WDB.

## References

1. S. Abiteboul, Querying semi-structured data. *Database Theory — ICDT'97, 6th International Conference*, Springer, 1997, 1–18
2. S. Abiteboul and V. Vianu, Queries and computation on the Web. *Database Theory — ICDT'97, 6th International Conference*, Springer, 1997
3. P. Aczel, *Non-Well-Founded Sets*, CSLI Lecture Notes. No **14** (1988)
4. P. Aczel, Final Universes of Processes, *9th Internat. Conf. on Math. Foundations of Programming Semantics*, ed. S.Brookes, et al. Springer LNCS **802** (1994), 1-28
5. P. Aczel, Lectures on Semantics: The initial algebra and final coalgebra perspectives, *Logic of Computation*, ed. H. Schwichtenberg, Springer (1997)
6. J.K. Barwise, *Admissible Sets and Structures*. Springer, Berlin, 1975
7. P. Buneman, S. Davidson, G. Hillebrand, D. Suciu, A query Language and Optimisation Techniques for Unstructured Data. *Proc. of SIGMOD*, San Diego, 1996
8. L. Cardelli, Semistructured Computation, Febr. 1, 2000; and other papers on mobile ambient calculi available via `http://www.luca.demon.co.uk/`
9. E. Dahlhaus and J.A. Makowsky, The choice of programming primitives for SETL-like programming languages, Proceedings of ESOP'86 (European Symp. on Programming, Saarbrüken, March 17-19, 1986), B. Robinet and R. Wilhelm eds., LNCS 213 (1986), 160–172.
10. E. Dahlhaus and J.A. Makowsky, Query languages for hierarchic databases, Information and Computation vol 101.1, (1992), pp. 1–32.
11. A. Davison, L. Sterling and S.W. Loke, *LogicWeb: LOGIC Programming for the World-Wide WEB*, `http://www.cs.mu.oz.au/~swloke/logicweb.html`
12. F. Durán, S. Eker, P. Lincoln, and J. Meseguer, Principles of Mobile Maude, LNCS, **1882**, , Springer-Verlag. 2000, 73–85.
13. R. Eschbach, U. Glässer, R. Gotzhein, and A. Prinz, On the Formal Semantics of Design Languages: A compilation approach using Abstract State Machines, Springer LNCS **1912** 2000, 131–151,
14. R. Fagin, Generalized first order spectra and polynomial time recognizable sets, *Complexity of Computations*, SIAM — AMS Proc. **7**, 1974, 43–73
15. M. Fisher and M. Wooldridge, On the Formal Specification and Verification of Multi-Agent Systems. *Intern. J. of Cooperative Information Systems*, 6(1), 1997
16. C. Fournet, G. Gonthier, J-J. Lévy, L. Maranget, and D. Rémy, A Calculus of Mobile Agents, *CONCUR'96*, Springer LNCS **1119** (1996), 406–421
17. R.O. Gandy, Set theoretic functions for elementary syntax. Proc. Symp. in Pure Math. Vol. **13**, Part II (1974), 103–126.
18. Y. Gurevich, Algebras of feasible functions. *FOCS* **24** (1983), 210–214
19. Y. Gurevich at al., Web-site on ASM: `http://www.eecs.umich.edu/gasm/`
20. N. Immerman, Relational queries computable in polynomial time, *Proc. 14th. ACM Symp. on Theory of Computing*, (1982) 147–152

21. R.B. Jensen, The fine structure of the constructible hierarchy. Ann. Math. Logic **4** (1972), 229–308.
22. A. Leontjev and V. Sazonov, Capturing LOGSPACE over Hereditarily-Finite Sets, *FoIKS'2000*, Springer LNCS **1762** (2000), 156–175
23. A. Lisitsa and V. Sazonov, Delta-languages for sets and LOGSPACE computable graph transformers. *Theoretical Computer Science* **175**, 1 (1997), 183–222
24. A. Lisitsa, and V. Sazonov, Bounded Hyper-set Theory and Web-like Data Bases, *KGC'97*, Springer LNCS **1289** (1997), 172–185
25. A. Lisitsa and V. Sazonov, Linear ordering on graphs, anti-founded sets and polynomial time computability, *Theoretical Computer Science* **224**, 1–2 (1999) 173-213
26. A.B. Livchak, Languages of polynomial queries. *Raschet i optimizacija teplotehnicheskih ob'ektov s pomosh'ju EVM*, Sverdlovsk, 1982, p. 41 (in Russian)
27. A.O. Mendelzon, G.A. Mihaila, T. Milo, Querying the World Wide Web, Int. J. on Digital Libraries 1(1): 54–67 (1997)
28. R. Milner, *Communication and Concurrency*, Prentice Hall, 1989
29. R. Milner, RJ. Parrow and D. Walker, A calculus of mobile processes, Parts 1–2, *Information and Computation*, 100(1), 1–66, 1992 (and other papers on Action calculi and the pi-calculus available via `http://www.cl.cam.ac.uk/users/rm135/`)
30. Nomadic Pict, `http://www.cl.cam.ac.uk/users/pes20/nomadicpict.html`
31. A. Sahuguet, B. Pierce and Val Tannen, Distributed Query Optimization: Can Mobile Agents Help? 2000, cf. `http://db.cis.upenn.edu/Publications/`
32. V. Sazonov, Polynomial computability and recursivity in finite domains. *Elektronische Informationsverarbeitung und Kybernetik*, **16** (7) (1980) 319–323
33. V. Sazonov, Bounded set theory and polynomial computability. *All Union Conf. Appl. Logic., Proc.* Novosibirsk, 1985, 188–191 (In Russian) Cf. also essentially extended English version with new results in *FCT'87,* LNCS **278** (1987), 391–397
34. V. Sazonov, Hereditarily-finite sets, data bases and polynomial-time computability. *TCS*, **119** Elsevier (1993), 187–214
35. V. Sazonov, A bounded set theory with anti-foundation axiom and inductive definability, *CSL'94* Springer LNCS **933** (1995) 527–541
36. V. Sazonov, On Bounded Set Theory. Invited talk on the *10th International Congress on Logic, Methodology and Philosophy of Sciences, in Volume I: Logic and Scientific Method*, Kluwer Academic Publishers, 1997, 85–103
37. V. Sazonov, Web-like Databases, Anti-Founded Sets and Inductive Definability, *Programmirovanie*, 1999, N5, 26-43 (In Russian; cf. also English version of this J.).
38. V. Sazonov and Yu. Serdyuk, Experimental implementation of set-theoretic query language Delta to Web-like databases (in Russian), *Programmnye Systemy, Teoreticheskie Osnovy i prilozhenija,* RAN, Institut Programmnyh System, Moskva, Nauka, Fizmatlit, 1999
39. Yu. Serdyuk, Partial Evaluation in a Set-Theoretic Query Language for WWW, *FoIKS'2000*, Springer LNCS **1762** (2000), 260–274
40. D. Suciu, Distributed Query Evaluation on Semistructured Data, 1997. Available from `http://www.research.att.com/~suciu`
41. D. Suciu, Query decomposition and view maintenance for query languages for unstructured data, *Proc. of the Internat. Conf. on Very Large Data Bases*, 1996
42. M.Y. Vardi, The complexity of relational query languages. *Proc. of the 14th. ACM Symp. on Theory of Computing*, (1982) 137–146