Hyperset Approach to Semi-structured Databases

Computation of Bisimulation in the Distributed Case

Richard Molyneux[1] and Vladimir Sazonov[2]

[1] Department of Computer Science, University of Liverpool, Liverpool, molyneux@liv.ac.uk
[2] Department of Computer Science, University of Liverpool, Liverpool, sazonov@liv.ac.uk

Abstract. We will briefly describe the recently implemented hyperset approach to semi-structured or Web-like and possibly distributed databases with the query system available online at http://www.csc.liv.ac.uk/~molyneux/t/. As this approach is crucially based on the bisimulation relation, the main stress in this paper is on its computation in the distributed case by using a so called bisimulation engine and local approximations of the global bisimulation relation.

1 Introduction

Unlike the more widely known traditional view on semi-structured data (SSD) as labelled graphs or as text based XML documents [1], the hyperset approach discussed in this paper assumes an abstract data model where such data is considered as set of sets of sets, etc. All elements of such sets, which are themselves sets, are assumed to have labels. Labels are the proper carriers of the atomic information, whereas sets just organise this information. Moreover, non-well-founded sets or hypersets allowing cyclic, as well as nested structure adopted in this approach are, in fact, a known extension of classical set theory.

Hyperset data model and graphs. The set theoretical view is, in fact, closely related with the graph view on semi-structured data by the evident and natural correspondence between labelled set equations like $s = \{l_1 : s_1, \ldots, l_n : s_n\}$, with s and s_i set names, and "fork" graph fragments determined by labelled directed edges outgoing from s: $s \xrightarrow{l_1} s_1, \ldots, s \xrightarrow{l_n} s_n$. This way a system of such (labelled) set equations can generate arbitrary directed graph (possibly with cycles), and vice versa. All sets and graphs are assumed to be finite. Another way to view such graphs is thinking on set names as analogies of URLs (Web addresses) and on the labels l_i as marking hyperlinks $s \xrightarrow{l_i} s_i$ between these URLs. This analogy also leads to the term Web-like database (WDB) [12] for any such system of set equations (and corresponding graph). "Graph" is essentially synonymous to the "web", and "semi-structured" usually means "structured as a graph". Semi-structured databases arose also essentially due to Web [1] as

the possibility of easy accessing various data having no unique and uniform structure. Moreover, the analogy with the WWW can be further supported by allowing system of set equations to be distributed amongst many WDB files, not necessarily in one computer, where a set name s participating in one file as $\ldots = \{\ldots, l : s, \ldots\}$ can possibly refer to other file in which s is described by appropriate set equation $s = \{l_1 : s_1, \ldots, l_n : s_n\}$. This assumes distinguishing between simple set names and full set names involving the URL of appropriate WDB file. The idea of distributed WDB seems inevitable and reasonable in the age of Internet (and Intranets) as it is unpractical and illogical to have all set equations of a big WDB to be contained in one file and in one computer.

Hypersets and relational databases. Before going further in describing hyperset approach, it makes sense to note the analogy of a set $s = \{l_1 : s_1, \ldots, l_n : s_n\}$ with a relational record where it is assumed the restriction that the fields or attributes $l_1, \ldots, l_n$ are all different. Thus, a set of such records with the same list of attributes l_i constitute a relation. Then, the possible distributed character of such data mentioned above would correspond, for example, to a heterogeneous system of relational databases as a very special case of arbitrary semi-structured hyperset data in the form of a system of such equations. Even if the same could be represented in terms of a (distributed) graph, the set theoretic approach (assuming considering arbitrary sets s of sets s_i, etc.) is closer to the spirit of relational (or nested relational) databases having essentially the same logical background which cannot be said so straightforwardly concerning the purely graph theoretical view. Also, the general hyperset query language Δ discussed below would potentially allow to query arbitrary heterogeneous distributed relational databases in a quite natural and logically coherent way. In fact, the great success of relational databases arose due to their highly transparent logical and set theoretical nature, and we develop the same set theoretical paradigm having, as we believe, good theoretical and practical potential.

Equality between hypersets. The concept of equality makes the main and crucial difference between graph nodes and hypersets. Hyperset view assumes that two set names (graph nodes) can denote the same abstract hyperset as the order of elements and repetitions—occasional *redundancies* in the data—should not play a role in the abstract meaning of these data. For example, given set equations

```
BibDB = {book:b,paper:p}
    b = {author:"Jones",title:"Databases"}
    p = {author:"Jones",title:"Databases"},
```

we should conclude that `b = p`.[3] Mathematically, this means that two set names (graph nodes) which are *bisimilar* or *informationally equivalent* denote the same

[3] It might be either intended or not by the designer of this database that the same publication is labelled both as a `book` and as a `paper`. Of course, this WDB can also be redesigned by replacing `book` and `paper` with `publication` and by adding elements `type:"book"` to `b` and `type:"paper"` to `p` thereby distinguishing `b` and `p`.

hypersets (see also [3, 4]). That is, possible redundancies and order in set equations should be ignored (or eliminated). In this sense the hyperset approach is a kind of graph approach where graph nodes are considered up to bisimulation. Anyway, the main style of thought implied here is rather a set theoretical one, particularly in writing queries to such databases. The corresponding set theoretic query language Δ will be briefly described, which was recently implemented [15] and now is fully functioning at least as a demo query system available online [7].

On the other hand, pure graph approach to SSDB such as [2] assumes that different graph nodes are considered as different "objects" or "object identities". Also in XML the predefined order of data is considered as essential (see e.g. [5] on querying XML and tree data). In the literature we find only one approach via the query language UnQL [6] which is most close to the hyperset one where queries to graph databases are also required to be bisimulation invariant. However, it is argued in [18], where UnQL is imitated by our query language Δ, that the former is still more graph rather than set theoretic approach because the graphs considered in UnQL have so called input and output nodes (which made the embedding of UnQL into Δ rather complicated and not very natural as these multiple inputs and outputs conflict with the hyperset view). Anyway, Δ has quite sufficient expressive power for this imitation. Also the mere bisimulation invariance used both in UnQL and Δ is only a half-story. An essential theoretical advantage of pure set theoretic query language Δ and its versions consists in precise characterisation of their expressive power in terms of PTIME [14, 17] and (N/D)LOGSPACE [11, 13].

As the bisimulation relation is crucial to this approach, and because its computation is in general sufficiently complicated, especially in the case of a distributed WDB, it is therefore highly important to find a practical way of dealing with bisimulation, particularly in the extremal distributed case. We suggest to compute the (global) bisimulation relation by using its local approximations computed locally. The computation of the global bisimulation is being done in background time by means of the so called centralised bisimulation engine, which requests local approximations of the bisimulation relation to compute the global bisimulation relation. This work to support computation of the bisimulation relation may be a permanent process as the (global) WDB can be locally updated and this should be repeatedly taken into account. Meanwhile, if a user runs a query q involving the set theoretic equality $x = y$, the query system asks the centralised bisimulation engine whether it already knows the answer to the question "x=y?". If not, the query system tries to compute it itself. But as soon as the engine will get the answer, it will send it to the query system which probably has not computed it yet. In this way, query evaluation involving equality becomes more realistic in the distributed case.

Note that there are known some approaches to efficient computing bisimulation relation such as [8–10] which should also be taken into account, although they do not consider the distributed case as we do. Of course, the distributed feature is not an innate one or belonging exclusively to the hyperset approach. But being a natural and in a sense an extreme possibility in general this makes

bisimulation relation—a really inherent to this approach—particularly challenging implementation task. We intend to show (and this paper and experiments in [15] are only a first step) that potentially even in this distributed case hyperset approach could be sufficiently realistic.

2 Hyperset Query Language Δ

The abstract syntax of the hyperset query language Δ is as follows:

$$\begin{aligned}\langle \Delta\text{-term}\rangle ::= {} & \langle \text{set variable or constant}\rangle \mid \emptyset \mid \{l_1 : a_1, \ldots, l_n, a_n\} \mid \bigcup a \mid \mathsf{TC}(a) \mid \\ & \{l : t(x,l) \mid l : x \in a \;\&\; \varphi(x,l)\} \mid \mathsf{Rec}\; p.\{l : x \in a \mid \varphi(x,l,p)\} \mid \mathsf{Dec}(a,b) \\ \langle \Delta\text{-formula}\rangle ::= {} & a = b \mid l_1 = l_2 \mid l_1 < l_2 \mid l_1\; R\; l_2 \mid l : a \in b \mid \varphi \;\&\; \psi \mid \varphi \vee \psi \mid \neg\varphi \mid \\ & \forall l : x \in a.\varphi(x,l) \mid \exists l : x \in a.\varphi(x,l)\end{aligned}$$

Here we denote: $a, b, \ldots$ as (set valued) Δ-terms; $x, y, z, \ldots$ as set variables; l, l_i as label values or variables (depending on the context); $l : t(x,l)$ is any l-labelled Δ-term t possibly involving the label variable l and the set variable x; and φ, ψ as (boolean valued) Δ-formulas. Note that labels l_i participating in the Δ-term $\{l_1 : a_1, \ldots, l_n : a_n\}$ need not be unique, that is, multiple occurrences of labels are allowed. This means that we consider arbitrary sets of labelled elements rather than records or tuples of a relational table where l_i serve as names of fields (columns). Label and set variables l, x, p of quantifiers, collect, and recursion constructs (see the descriptions below) should not appear free in the bounding term a (denoting a finite set). Otherwise, these operators may become actually unbounded and thus, in general, non-computable in finite time.

More details on the meaning of the above constructs and on the implemented version of Δ can be found e.g. in [18, 16, 15, 7]. Many constructs of Δ should be quite evident. Here we only mention briefly that

- $\bigcup$ means *union* of a set of sets,
- TC means *transitive closure* of a set (elements, elements of elements, etc.),
- the *collection* operator $\{l : t(x,l) \mid l : x \in a \;\&\; \varphi(x,l)\}$ denotes the set of all labelled elements $l : t(x,l)$ such that $l : x \in a \;\&\; \varphi(x,l)$ holds,
- the *recursion* operator $\mathsf{Rec}\; p.\{l : x \in a \mid \varphi(x,l,p)\}$ defines iteratively a set π satisfying the identity $\pi = \pi \cup \{l : x \in a \mid \varphi(x,l,\pi)\}$,
- the *decoration* operator $\mathsf{Dec}(g,v)$ denotes a unique hyperset corresponding to the vertex v of a graph g. Here g is any set of labelled ordered pairs $l : \{fst : x, snd : y\}$ understood as graph edges $x \xrightarrow{l} y$, and v is one of these x, y participating in such pairs. Note that x, y, v are considered here as arbitrary hypersets. Dec can also be naturally called *plan performance operator*. Given a graphical plan consisting of g and v, it constructs a unique hyperset $\mathsf{Dec}(g,v)$ according to this plan.
- $<$ and R denote alphabetic ordering on labels and, respectively, substring relation.

2.1 Rough Description of the Operational Semantics of Δ [18]

Operational semantics of Δ is required to implement this language. The working query system is described in detail in [16, 15, 7] where examples of queries can be found and, in fact, run on the implementation available online.

Consider any set or boolean query q in Δ which involves no free variables and whose participating set names (constants) are taken from the given WDB system of set equations. Resolving q consists in the following two macro steps:

- **Extending** this system by the new equation $res = q$ with res a fresh (i.e. unused in WDB) set or boolean name, and
- **Simplifying** the extended system $\mathrm{WDB}_0 = \mathrm{WDB} + (res = q)$ by some (quite natural) reduction steps $\mathrm{WDB}_0 \rhd \mathrm{WDB}_1 \rhd \ldots \rhd \mathrm{WDB}_{res}$ until it will contain only flat bracket expressions as the right-hand sides of the equations or the truth values *true* or *false* (if the left-hand side is boolean name).

After simplification is complete, these set equations will contain no complex set or boolean queries (like q above). In fact, the resulting version WDB_{res} of WDB will consist (alongside the old equations of the original WDB) of new set equations (new, possibly auxiliary set names equated to flat bracket expressions) and boolean equations (boolean names equated to boolean values, *true* or *false*). We cannot go here into further details, except saying that in the case of a set query q the simplified version $res = \{\ldots\}$ of the equation $res = q$ will give the resulting value for q. We only will consider below the case of equality query $x = y$ (with the value *true* or *false*) in terms of the bisimulation relation $x \approx y$.

3 Bisimulation

Assume WDB is represented as a system of set equations $\bar{x} = \bar{b}(\bar{x})$ where $\bar{x}$ is a list of set names $x_1, \ldots, x_k$ and $\bar{b}(\bar{x})$ is the corresponding list of bracket expressions (for simplicity, "flat" ones). The set of all set names $x_1, \ldots, x_k$ of the given WDB is also denoted as *SNames*. Visually equivalent representation can be done in the form of labelled directed graph, where labelled edges $x_i \xrightarrow{label} x_j$ correspond to the set memberships $label : x_j \in x_i$ meaning that the equation for x_i has the form $x_i = \{\ldots, label : x_j, \ldots\}$. In this case we also call x_j a child of x_i. Note that this particular, *concrete* usage of the membership symbol $\in$ as relation between set names or graph nodes is non-traditional but very close to the traditional set theoretic membership relation between *abstract* (hyper)sets, hence we decided not to introduce a new kind of membership symbol here. For the simplicity of our description below labels are ignored as they would not affect essentially the nature of our considerations.

3.1 Hyperset equality and the problem of efficiency

One of the key points of our approach is the interpretation of WDB-graph nodes as set names $x_1, \ldots, x_k$ where different nodes x_i and x_j can, in principle, denote

the same (hyper)set, $x_i = x_j$. This particular notion of equality between nodes is defined by the bisimulation relation denoted also as $x_i \approx x_j$ (to emphasise that set names can be syntactically different, but denote the same set) which can be computed by the appropriate recursive comparison of child nodes or set names. Thus, in outline, to check bisimulation of two nodes we need to check bisimulation between some children, grandchildren, and so on, of the given nodes, i.e. many nodes could be involved. If the WDB is distributed amongst many WDB files and remote sites then downloading the relevant WDB files might be necessary in this process and will take significant time. (There is also the analogous problem with the related transitive closure operator TC whose efficient implementation in the distributed case can require similar considerations.) So, in practice the equality relation for hypersets seems intractable, although theoretically it takes polynomial time with respect to the size of WDB. Nevertheless, we consider that the hyperset approach to WDB based on bisimulation relation is worth implementing because it suggests a very clear and mathematically well-understood view on semi-structured data and the querying of such data. Thus, the crucial question is whether the problem of bisimulation can be resolved in any reasonable and practical way. One approach related with the possibly distributed nature of WDB and showing that the situation is manageable in principle is outlined below.

3.2 Computing bisimulation relation $\approx$ over WDB

Bisimulation relation can be computed (and thereby defined) by deriving negative ($\not\approx$) bisimulation facts by means of the following recursive rule:

$$x \not\approx y :- \exists x' \in x \forall y' \in y (x' \not\approx y') \vee \exists y' \in y \forall x' \in x (x' \not\approx y') \qquad (1)$$

where initial negative facts can be obtained by the partial case of this tule:

$$x \not\approx y :- (x = \emptyset \ \& \ y \neq \emptyset) \vee (y = \emptyset \ \& \ x \neq \emptyset).$$

This means that any set described as empty one is nonbisimilar to any set described as non-empty in the WDB. The evident dual rule to (1) can also be used for deriving positive bisimulation facts. However in principle this is unnecessary as such facts will be obtained, anyway, at the moment of stabilisation in the derivation process by using only (1) as there are only finitely many of WDB set names in *SNames*.

Equivalently, $\not\approx$ is the least relation satisfying (1), and its positive version $\approx$ is the largest relation satisfying

$$x \approx y \Rightarrow \forall x' \in x \exists y' \in y (x' \approx y') \ \& \ \forall y' \in y \exists x' \in x (x' \approx y'). \qquad (2)$$

It is well-known known that bisimulation $\approx$ is an equivalence relation which is completely coherent with hyperset theory as it is fully described in [3, 4] for the pure case, and this fact extends easily to the labelled case. It is by this reason that the bisimulation relation $\approx$ between set names can be considered as an equality relation $=$ between corresponding abstract hypersets.

3.3 Local Approximations of $\approx$

Now, let a non-empty set $L \subseteq SNames$ of "local" vertices (set names) in a graph WDB (a system of set equations) be given, where *SNames* is the set of all WDB vertices (set names). Let us also denote by $L' \supseteq L$ the set of all "almost local" set names participating in the set equations for each set name in L both at left and right-hand sides. Considering the graph as a WDB distributed among many *sites*, L plays the role of (local) set names defined by set equations in some (local) WDB files of one of these sites. Then $L' \setminus L$ consists of non-local set names which, however, participate in the local WDB files, have defining equations in other (possibly remote) sites of the given WDB. Non-local (full) set names can be recognised by their URLs as different from the URL of the given site.

We will consider *derivation rules* of the form $xRy :- \ldots R \ldots$ for two more relations $\approx^L_-$ and $\approx^L_+$ over *SNames*:

$$\approx^L_- \subseteq \approx \subseteq \approx^L_+ \quad \text{or, rather, their negations} \quad \not\approx^L_+ \subseteq \not\approx \subseteq \not\approx^L_- \tag{3}$$

defined formally on the whole WDB graph (however, we will be mainly interested in the behaviour of $\approx^L_-$ and $\approx^L_+$ on L). We will usually omit the superscript L as we currently deal mainly with one L, so no ambiguity can arise.

3.4 Defining the Local Upper Approximation $\approx^L_+$ of $\approx$

Let us define the relation $\not\approx_+ \subseteq SNames^2$ by derivation rule

$$x \not\approx_+ y :- x, y \in L \;\&\; [\exists x' \in x \forall y' \in y(x' \not\approx_+ y') \vee \ldots]. \tag{4}$$

Here and below "..." represents the evident symmetrical disjunct (or conjunct). Thus the premise (i.e. the right-hand side) of (4) is a *restriction* of that of (1). It follows by induction on the length of derivation of the $\not\approx_+$-facts that,

$$\not\approx_+ \subseteq \not\approx, \quad \approx \subseteq \approx_+, \tag{5}$$

$$x \not\approx_+ y \Rightarrow x, y \in L, \tag{6}$$

Let us also consider another, "more local" version of the rule (4):

$$x \not\approx_+ y :- x, y \in L \;\&\; [\exists x' \in x \forall y' \in y(x', y' \in L \;\&\; x' \not\approx_+ y') \vee \ldots]. \tag{7}$$

It defines the same relation $\not\approx_+$ because in both cases (6) holds implying that the right-hand side of (7) is equivalent to the right-hand side of (4). The advantage of (4) is its formal simplicity whereas that of (7) is its "local" computational meaning. From the point of view of distributed WDB with L one of its local sets of vertices/set names (corresponding to one of the sites of the distributed WDB), we can derive $x \not\approx_+ y$ for local x, y via (7) by looking at the content of local WDB files only. Indeed, participating URLs (full set names) $x' \in x$ and $y' \in y$, although likely non-local names ($\in L' \setminus L$), occur in the locally stored WDB files with local URLs x and $y \in L$. However, despite the possibility that x' and y' can be in general non-local, we will need to use in (7) the facts of the kind $x' \not\approx_+ y'$ derived on the previous steps for local $x', y' \in L$ only. Therefore,

Note 1 (Local computability of $x \not\approx_+ y$). For deriving the facts $x \not\approx_+ y$ for $x, y \in L$ by means of the rule (4) or (7) we will need to use the previously derived facts $x' \not\approx_+ y'$ for set names x', y' from L only, and additionally we will need to use set names from a wider set L' (available, in fact, also locally)[4]. In this sense, the derivation of all facts $x \not\approx_+ y$ for $x, y \in L$ can be done locally and does not require downloading of any external WDB files. (In particular, facts of the form $x \not\approx_+ y$ or $x \approx_+ y$ for set names x or y in $L' \setminus L$ present no interest in such derivations.)

The upper approximation $\approx_+$ (on the whole WDB graph) can be equivalently characterised as the largest relation satisfying any of the following (equivalent) implications for all graph vertices x, y:

$$\begin{aligned} x \approx_+ y \Rightarrow x \notin L \vee y \notin L \vee [\forall x' \in x \exists y' \in y (x' \approx_+ y') \;\&\; \ldots] \\ x \approx_+ y \;\&\; x, y \in L \Rightarrow [\forall x' \in x \exists y' \in y (x' \approx_+ y') \;\&\; \ldots] \end{aligned} \tag{8}$$

It is easy to show that the set of relations $R \subseteq SNames^2$ satisfying (8) (in place of $\approx_+$) **(i)** contains the identity relation $=$, **(ii)** is closed under unions (thus the largest $\approx_+$ does exist), and **(iii)** is closed under taking inverse. Evidently, any ordinary (global) bisimulation relation $R \subseteq SNames^2$ (that is, a relation satisfying (2)) satisfies (8) as well. For any $R \subseteq L^2$ the converse also holds: if R satisfies (8) then it is actually a global bisimulation relation (and $R \subseteq \approx$). It is also easy to check that **(iv)** relations $R \subseteq L^2$ satisfying (8) are closed under compositions. It follows from **(i)** and **(iii)** that $\approx_+$ is reflexive and symmetric. Over L, the relation $\approx_+$ (that is the restriction $\approx_+ \restriction L$) is also transitive due to **(iv)**. Therefore, $\approx_+$ is an *equivalence relation* on L. (In general, $\approx_+$ cannot be an equivalence relation on the whole graph due to (6) if $L \neq SNames$.) Moreover, any $x \notin L$ is $\approx_+$ to all vertices (including itself).

3.5 Defining the local lower approximation $\approx_-^L$ of $\approx$

Consider the derivation rule for the relation $\not\approx_- \subseteq SNames^2$:

$$\begin{aligned} x \not\approx_- y :- [(x \notin L \vee y \notin L) \;\&\; x \neq y] \vee \\ [\exists x' \in x \forall y' \in y (x' \not\approx_- y') \vee \ldots] \end{aligned} \tag{9}$$

which can also be equivalently replaced by two rules:

$$\begin{aligned} &x \not\approx_- y :- (x \notin L \vee y \notin L) \;\&\; x \neq y - \text{“a priori knowledge”}, \\ &x \not\approx_- y :- \exists x' \in x \forall y' \in y (x' \not\approx_- y') \vee \ldots \;. \end{aligned} \tag{10}$$

[4] This is the case when $y = \emptyset$ but there exists according to (7) an x' in x which can be possibly in $L' \setminus L$ (or similarly for $x = \emptyset$). When $y = \emptyset$ then, of course, there are no suitable witnesses $y' \in y$ for which $x' \not\approx_+ y'$ hold. Therefore, only the existence of some x' in x plays a role here.

Thus, in contrast to (4), this is a *relaxation* or an *extension* of the rule (1) for $\not\approx$. It follows that

$$\not\approx \subseteq \not\approx_- \ (\approx_- \subseteq \approx).$$

It is also evident that

$$\begin{aligned} &\text{any } x \notin L \text{ is } \not\approx_- \text{ to all vertices different from } x, \\ &x \approx_- y \ \& \ x \neq y \Rightarrow (x, y \in L). \end{aligned}$$

The latter means that $\approx_-$ (which is an equivalence relation on *SNames* and hence on L as it is shown below) is non-trivial only on the local set names. Again, like for $\not\approx_+$, we can conclude from the above considerations that,

Note 2 (Local computability of $x \not\approx_- y$). We can compute the restriction of $\not\approx_-$ on L locally: to derive $x \not\approx_- y$ for $x, y \in L$ with $x \neq y$ (taking into account reflexivity of $\approx_-$) by (9) we need to use only $x', y' \in L'$ (by $x' \in x$ and $y' \in y$) and already derived facts $x' \not\approx_- y'$ for $x', y' \in L, x \neq y$, as well as the facts $x' \not\approx_- y'$ for x' or $y' \in L' \setminus L$, $x' \neq y'$ following from the "a priori knowledge" (10).

The lower approximation $\approx_-$ can be equivalently characterised as the largest relation satisfying

$$x \approx_- y \Rightarrow (x, y \in L \vee x = y) \ \& \ (\forall x' \in x \exists y' \in y (x' \approx_- y') \ \& \ \ldots).$$

Evidently, $=$ (substituted for $\approx_-$) satisfies this implication. Relations R satisfying this implication are also closed under unions and taking inverse and compositions. It follows that $\approx_-$ is reflexive, symmetric and transitive, and therefore an *equivalence relation over the whole WDB graph*, and hence *on its local part L*.

Thus, both approximations $\approx_+^L$ and $\approx_-^L$ to $\approx$ are computable "locally". Each of them is defined in a trivial way outside of L, and the computation requires only knowledge at most on the L'-part of the graph. In fact, only edges from L to L' are needed, everything being available locally.

3.6 Using local approximations to aid computation of the global bisimulation

Now, assume that the set *SNames* of all set names (nodes) of a WDB is disjointly divided into a family of local sets L_i, for each "local" site $i \in I$ with local approximations $\approx_+^{L_i}$ and $\approx_-^{L_i}$ to the global bisimulation relation $\approx$ computed locally. Now the problem is how to compute the global bisimulation relation $\approx$ with the help of many its local approximations $\approx_+^{L_i}$ and $\approx_-^{L_i}$ in all sites $i \in I$.

Granularity of sites. However, for simplicity of implementation and testing the above idea and also because this is reasonable in itself we will redefine the scope of i to a smaller granularity. Instead of taking i to be a site, consisting of many WDB files, we will consider that each i itself is a name of a single WDB file

file$_i$. More precisely, i is considered as the URL of any such a file. This will not change the main idea of implementation of the bisimulation Oracle on the basis of using local information for each i. That is, we reconsider our understanding of the term local – from being *local to a site* to *local to a file*. Then L_i is just the set of all (full versions of) set names defined in file i (left-hand sides of all set equations in this file). Evidently, so defined sets L_i are disjoint and cover the class *SNames* of all (full) set names from the WDB considered.

Then the relations $\approx_+^{L_i}$ and $\approx_-^{L_i}$ should be automatically computed and maintained as the current local approximations for each WDB file i each time this file is updated. In principle a suitable tool is necessary for editting (and maintaining) WDB, which would save a WDB file i and thereby generate and save the approximation relations $\approx_+^{L_i}$ and $\approx_-^{L_i}$ automatically.

In general, we can reasonably use even more levels of locality distributing the workload between many servers of various levels acting in parallel.

Local approximations giving rise to global bisimulation facts. It evidently follows from (3) that

- *each positive local fact of the form* $x \approx_-^{L_i} y$ *gives rise to the fact* $x \approx y$*, and*
- *each negative local fact of the form* $x \not\approx_+^{L_i} y$ *gives rise to the fact* $x \not\approx y$*.*

Let $\approx^{L_i}$ (without subscripts $+$ or $-$) denote the set of positive and negative facts for set names in L_i on the global bisimulation relation $\approx$ obtained by these two clauses. This set of facts $\approx^{L_i}$ is called the *local simple approximation set* to $\approx$ for the file (or site) i. Let the *local Oracle* LO_i just answer *"Yes"* ("$x \approx y$"), *"No"* ("$x \not\approx y$") or *"Unknown"* to questions $x \stackrel{?}{\approx} y$ for $x, y \in L_i$ according to $\approx^{L_i}$.

In the case of i considered as a site (rather than a file), LO_i can have delays when answering *"Yes"* ("$x \approx y$") or *"No"* ("$x \not\approx y$") because LO_i should rather compute $\approx^{L_i}$ itself and find out in $\approx^{L_i}$ answers to the questions asked which takes time. But, if i is understood just as a file saved together with all the necessary information on local approximations at the time of its creation then LO_i can submit the required answer and, additionally, all the other facts it knows at once (to save time on possible future communications).

Therefore, a centralised Internet server (for the given distributed WDB) working as the (global) Oracle or *Bisimulation Engine*, which derives positive and negative ($\approx$ and $\not\approx$) global bisimulation facts, can do this by a natural algorithm based on the derivation rule (1), additionally asking (when required) various local Oracles LO_i concerning $\approx^{L_i}$. That is, the algorithm based on (1) and extended to exploit local simple approximations $\approx^{L_i}$ should, in the case of the currently considered question $x \stackrel{?}{\approx} y$ with $x, y \in L_i$ from the same site/WDB file i[5], additionally ask the oracle LO_i whether it already knows the answer (as described in the above two items). If the answer is known, the algorithm should just use it (as it was in fact derived in a local site). Otherwise (if LO_i does not

[5] $x, y \in L_i$ iff the full versions of set names x, y have the same URL i.

know the answer or x, y do not belong to one L_i – that is, they are "remote" one from another), the global Oracle should work according to (1) by downloading the necessary set equations, making derivation steps, asking the local Oracles again, etc. Thus, local approximations serve as auxiliary local Oracles LO_i helping the global Oracle.

4 Bisimulation Engine (the Oracle)

The idea is to have a centralised service providing answers to bisimulation question which would improve query performance (for those queries exploiting set equality). This service could be named *Bisimulation Engine* or just the *Oracle*. The goal of such bisimulation engine would consist in:

- **Answering bisimulation queries** asked by (any of possible copies[6] of) Δ-query system via appropriate protocol.
- **Computing bisimulation** by deriving bisimulation facts in background time, and strategically prioritising bisimulation questions posed by the Δ-query systems by temporary changing the fashion of the background time work in favour of resolving these particular questions.
- **Exploiting local approximations** $\approx_{-}^{L_i}$, $\approx_{+}^{L_i}$ and $\approx^{L_i}$ corresponding to WDB servers/files i of a lower level of locality to assist in the computation of bisimulation.
- **Maintaining cache of set equations** downloaded in the previous steps. These set equations may later prove to be useful in deriving new bisimulation facts, saving time on downloading of already known equations.

Moreover, it is reasonable to make the query system adopt its own "lazy" prioritisation strategy while working on a query q. This strategy consists of sending bisimulation subqueries of q to the Oracle but not attempting to resolve them in the case of the Oracle's answer "Unknown". Instead of such attempts, the query system could try to resolve other subqueries of the given query q until the resolution of the bisimulation question sent to the Oracle is absolutely necessary. The hope is that before this moment the bisimulation engine will have already given a definite answer.

However these useful prioritisation features have not yet been implemented. Moreover, currently we have only a simplified imitation of bisimulation engine which resolves all possible bisimulation questions for the given WDB in some predefined standard order without any prioritisation and answers these questions in a definite way when it has derived the required information. The Oracle, while doing its main job in background time, should only remember all the pairs (client, question) for questions asked by clients and send the definite answer to the corresponding client when it is ready.

More detailed algorithm of such a Bisimulation Engine and some encouraging experiments and artificial examples of a WDBs for these experiments were described in detail in [15] which we briefly present below. They show the benefit

[6] There could be many users running each their own copy of the Δ-query system.

both of background work of this engine and of using local approximations to the bisimulation relation on the WDB.

Determining the benefit of background work by the bisimulation engine on query performance. For 51 set names distributed over 10 WDB files, connected in chains and an isomorphic copy of the same it was shown that querying $x \approx x'$ for the root nodes with a delay d (after the Bisimulation Engine started working) has performance time with exponential decay depending on d. Without the Bisimulation Engine execution of $x \approx x'$ would take about 20 seconds, but with using it after 5 seconds of delay it takes about 8 seconds, and after 20 seconds of delay it takes 10 milliseconds. As Bisimulation Engine is assumed to work permanently, the delay time should not count in the overall performance.

Determining the benefit of exploiting local approximations by the Bisimulation Engine on query performance. For two isomorphic chains $x_1 \to x_2 \to \ldots \to x_n$ and $x'_1 \to x'_2 \to \ldots \to x'_n$ the bisimulation $x_1 \approx x'_1$ takes 112 and 84 minutes for $n = 70$ with the Bisimulation Engine not exploiting local approximations and, respectively, without using the Bisimulation Engine at all, and it takes 40 seconds with using it and local approximations to these two chains. The dependence on n was also experimentally measured. Here we assume $d = 0$, so the difference $112 - 84 = 34$ min is the additional expense of $\sim 70^2$ communications with the Bisimulation Engine instead of getting benefit from two files with $\approx^L$ and $\approx^{L'}$.

Determining the benefits of background work by the bisimulation engine exploiting local approximations. For three chain files $x \to x_1 \to x_2 \to \ldots \to x_{20}$, $y_1 \to y_2 \to \ldots \to y_{20}$, $z_1 \to z_2 \to \ldots \to z_{20}$ and two additional external edges $x \to y_1$ and $x \to z_1$ + an isomorphic copy of the same checking $x \approx x'$ shows that the execution time sharply falls in from about 17 minutes of pure querying to 180 milliseconds (and then to 10 ms) for the delay $d = 5$ seconds.

Of course, these are rather oversimplified experiments. We also did not take into account other works on efficient computation of bisimulation for non-distributed case [8–10]. But it is evident that the general qualitative picture should remain the same. Also more realistic large scaled experiments are required.

Note that the current implementation of the hyperset language Δ [7] does not use yet any bisimulation engine. These experiments were implemented separately and only to demonstrate some potential benefits of using such an engine.

Conclusion

While the hyperset approach to semi-structured databases is very natural theoretically and even practically as example queries in [7, 15, 16] show, developing some efficient way of dealing with hyperset equality (bisimulation) is required. We have demonstrated one such approach based on local approximations to the global bisimulation relation and on the idea of background time computation somewhat similar to that of Web search engines. The experiments presented in [15] and above show that this idea is promising, however further experiments and

improvements should be done to make this approach more realistic. Another important topic to be considered is the impact of (possibly) dynamic updates of the WDB on the whole process, and also whether and under which conditions local updates can have small consequences. More general, the current implementation of the query language Δ [7] already working as a demo version and the language itself should be further improved/extended to make it more efficient and user friendly.

References

1. S. Abiteboul, P. Buneman, and D. Suciu. *Data on the Web — From Relations to Semi-structured Data and XML*. Morgan Kaufmann Publishers, 2000.
2. S. Abiteboul, D. Quass, J. McHugh, J. Widom, and J. L. Wiener. The Lorel query language for semistructured data. *International Journal on Digital Libraries*, 1(1):68–88, 1997.
3. P. Aczel. *Non-Well-Founded Sets*. CSLI, Stanford, CA, USA, 1988.
4. J. Barwise and L. Moss. *Vicious circles: on the mathematics of non-well-founded phenomena*. Center for the Study of Language and Information, 1996.
5. M. Benedikt, L. Libkin, and F. Neven. Logical definability and query languages over ranked and unranked trees. *ACM Transactions on Computational Logic (TOCL)*, 8(2):1–62, 2007.
6. P. Buneman, M. Fernández, and D. Suciu. UnQL: a query language and algebra for semistructured data based on structural recursion. *The VLDB Journal*, 9(1):76–110, 2000.
7. Delta-WDB Site. `http://www.csc.liv.ac.uk/~molyneux/t/`. 2008.
8. A. Dovier, C. Piazza, and A. Policriti. An efficient algorithm for computing bisimulation equivalence. *Theoretical Computer Science*, 311(1-3):221–256, 2004.
9. J.-C. Fernandez. An implementation of an efficient algorithm for bisimulation equivalence. *Science of Computer Programming*, 13(1):219–236, 1989.
10. P. Jancar and F. Moller. Techniques for decidability and undecidability of bisimilarity. In *CONCUR'99 Concurrency Theory*, pages 30–45, 1999.
11. A. Leontjev and V. Sazonov. Δ: Set-theoretic query language capturing logspace. *Annals of Mathematics and Artificial Intelligence*, 33(2-4):309–345, 2001.
12. A. Lisitsa and V. Sazonov. Bounded hyperset theory and web-like data bases. In *Proceedings of the Kurt Goedel Colloquium*, volume 1234, pages 178–188, 1997.
13. A. Lisitsa and V. Sazonov. Δ-languages for sets and LOGSPACE computable graph transformers. *Theoretical Computer Science*, 175(1):183–222, 1997.
14. A. Lisitsa and V. Sazonov. Linear ordering on graphs, anti-founded sets and polynomial time computability. *Theoretical Computer Science*, 224(1–2):173–213, 1999.
15. R. Molyneux. *Hyperset Approach to Semi-structured Databases and the Experimental Implementation of the Query Language Delta*. PhD thesis, University of Liverpool, Liverpool, England, 2008.
16. R. Molyneux and V. Sazonov. Hyperset/web-like databases and the experimental implementation of the query language delta - current state of affairs. In *ICSOFT 2007 Proceedings of the Second International Conference on Software and Data Technologies*, volume 3, pages 29–37. INSTICC, 2007.
17. V. Sazonov. Hereditarily-finite sets, data bases and polynomial-time computability. *Theoretical Computer Science*, 119(1):187–214, 1993.
18. V. Sazonov. Querying hyperset / web-like databases. *Logic Journal of the IGPL*, 14(5):785–814, 2006.