Querying Hyperset/Web-Like Databases

Vladimir Sazonov

University of Liverpool

V.Sazonov@csc.liv.ac.uk

http://www.csc.liv.ac.uk/~sazonov
The **goal** of this talk is in fact modest:

- to give a simple presentation of *hyperset approach* to semistructured (web-like) databases and of the corresponding *query language* $\Delta$,

- to answer the question on how *expressive power* of querying to semistructured databases can be characterised
  - in terms of *descriptive complexity*,
  - by defining *formal semantics of path expressions* in terms of $\Delta$,
  - by defining *formal semantics of other known languages* UnQL and UnCAL to semistructured data (based also on *structural recursion*) which are *most close to* $\Delta$. 
Expressive Power?

Some citations:

[S. Abiteboul, P. Buneman, D. Suciu, Data on the Web: From Relations to Semistructured Data and XML, 2000]:

“There are simple restructuring problems that defeat all the query languages we shall describe. Moreover, there is no accepted notion of completeness for semistructured data restructuring. Therefore, it is yet unclear what expressive power should capture a query language for semistructured data.”

[L. Cardelli, Semistructured Computation, 2000]:

“It should be noted that basic questions of expressive power for semistructured database query languages are still open.”
Expressive Power?

_Hyperset approach_ resolves this problem for semistructured databases represented as _finite graphs with labelled edges_ where _no ordering_ is assumed on the nodes.

— _Not_ the case of _XML_ where the data are ordered, although the ideas can evidently be applied to.

It consists in using the concept of

- *(finite) set* of sets of sets, etc., even _allowing cycles_, like in the simplest proper hyperset

\[ \Omega = \{\Omega\} \]

_in its full generality_ as a straightforward and transparent _abstract model of arbitrary data having no predefined structure._

- Abstract _hyperset theory_ with some applications is described in books [Aczel 1988] and [Barwise, Moss 1996].

- This is also a very natural way of _generalising the ordinary relational view to the case of semistructured databases_.

### Relational Databases as a Restricted Set-theoretic Approach to DB

<table>
<thead>
<tr>
<th>Relation Name</th>
<th>STUDENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attr&lt;sub&gt;1&lt;/sub&gt;</td>
<td>NAME</td>
</tr>
<tr>
<td>Attr&lt;sub&gt;2&lt;/sub&gt;</td>
<td>value&lt;sub&gt;1&lt;/sub&gt;</td>
</tr>
<tr>
<td>...</td>
<td>value&lt;sub&gt;i&lt;/sub&gt;</td>
</tr>
<tr>
<td>Attr&lt;sub&gt;N&lt;/sub&gt;</td>
<td>value&lt;sub&gt;N&lt;/sub&gt;</td>
</tr>
</tbody>
</table>

*Database state* is a finite *set* of relations.

\[
DB = \{R_1, R_2, \ldots, R_N\}
\]

*Relation* is a finite *set* of tuples (rows, records).

\[
\text{STUDENTS} = \{\text{student1, student2, student3}\}
\]

*Tuple* is also a finite *set* of *labelled “atomic” values*:

\[
\text{student1} = \{\text{NAME : I.Ivanov, BIRTH-DATE : 1981, DEPT : CS}\}
\]

Therefore relational DB is a *set of sets of sets* of *Nesting = 3*.

- Why not *arbitrary nesting*? (Nested relational databases).
- What about *cycling* (and, in general, *Web-like Databases*)?
  - CS may contain further “deep” information, say referring via a chain of “clicks” again to the STUDENTS table.

- Thus, we need also “cycling” sets, like \( \Omega = \{\Omega\} \) which exist in *non-well-founded set theory* and are also called *hypersets.*
Semistructured Databases (SSD): Graphs or Hypersets?

Navigational view on SSD/WWW/WDB as a graph

Fork or fan fragment of the whole graph:

Only edges outgoing from a given node are shown.

Corresponds to “mouse clicks” on the labels $l_i$ on the Web page with URL $u$.

The same set theoretically:

$$u = \{l_1 : u_1, \ldots, l_n : u_n\}$$

Thus, graph nodes represent (hyper)sets consisting of labelled elements.
Semistructured Databases (SSD): Graphs or Hypersets?

Navigational view on SSD/WWW/WDB leads to
- graph representation of data, and also to
- stressing on path expressions as the basic part of corresponding query languages such as Lorel or UnQL or XPath (the latter devoted to XML).

Hyperset view is
- more abstract, but
- completely consistent with navigational view, and
- bears ideas from set theory rather than from graphs.

Hypersets and graphs are inseparable, and serve as two levels of abstractions in hyperset approach:
- conceptual, or denotational level (hypersets), and
- computational level (graphs)
Hypersets represented by set equations

Consider any (recursive) system of set equations like

\[ s_5 = \{ l_1 : s_3, l_2 : s_5, l_2 : s_3 \} . \]

In general,

\[
\text{WDB} : \begin{cases}
    s_1 = B_1(s_1, \ldots, s_n), \\
    s_2 = B_2(s_1, \ldots, s_n), \\
    \ldots \\
    s_n = B_n(s_1, \ldots, s_n).
\end{cases}
\]

or in vector form:

\[
\text{WDB} : \{ \vec{s} = \vec{B}(\vec{s}) \} .
\]

This is actually a textual representation of an arbitrary finite directed graph with labelled edges where set names \( s_i \) serve as nodes.

Elements of the set \( s_5 = \{ l_1 : s_3, l_2 : s_5, l_2 : s_3 \} \) represent outgoing from \( s_5 \) edges labelled accordingly:

\[ s_5 \overset{l_1}{\rightarrow} s_3, \quad s_5 \overset{l_2}{\rightarrow} s_5, \quad s_5 \overset{l_2}{\rightarrow} s_3 \]
Hypersets represented by set equations

Antifoundation Axiom (AFA) of hyperset theory says that each such system of equations has a unique solution in the abstract universe of hypersets.

This way we achieve a natural level of abstraction making the hyperset approach to SSD/WDB very attractive.

In particular, two set names $s_i$ and $s_j$ could denote the same abstract hyperset,

$$s_i = s_j.$$

This means that in our database WDB $s_i$ and $s_j$ are informationally indistinguishable, or bisimular.

Otherwise, they are called informationally distinguishable, or non-bisimular, $s_i \neq s_j$. 
Hypersets represented by set equations

*Distinguishability* of two sets can be checked by the repeated applying the following two symmetric *derivation rules* (for simplicity without labels)

\[ s_i \neq s_j \iff \exists w \in s_i \forall v \in s_j (w \neq v) \]
\[ s_j \neq s_i \iff \exists w \in s_i \forall v \in s_j (w \neq v) \]

meaning that two sets \( s_i \) and \( s_j \) are different (distinguishable) if they *have different elements*. 
Hypersets represented by set equations

Knowing equality (bisimulation) relation allows for removing redundancies in the data:

\[
\begin{align*}
u &= \{a:u, a:v, b:v\} \\
v &= \{\}\quad u_1 &= \{a:u_2, b:v_1, a:v_2\} \\
v_1 &= \{\}\quad u_2 &= \{a:u_1, a:v_1, b:v_2\} \\
v_2 &= \{\}\quad v_1 &= \{\} \\
v_2 &= \{\}
\end{align*}
\]
Distributed WDB

WDB, as a *system of set equations*, can be

- *distributed* between several files,

- even on various *remote sites*.

These can be, for example, *HTML* files or *XML* files of special form which should be *hyperlinked*.

If some set name is described in another file the link should lead to the appropriate set equation in that file.
Hyperset Query Language $\triangle$

the basic [Gandy 74] or rudimentary [Jensen 72] fragment

Define inductively $\triangle$-formulas and $\triangle$-terms by the clauses

$\langle \triangle\text{-term} \rangle ::= \langle \text{set variable or constant} \rangle | \emptyset | \{l_1 : a_1, \ldots, l_n, a_n\} |
\bigcup a \mid TC(a) \big| \{l : t(x, l) \mid l : x \in a \& \varphi(x, l)\}$

$\langle \triangle\text{-formula} \rangle ::= a = b \mid l_1 R l_2 \mid l_0 : a \in b \mid \varphi \& \psi \mid \varphi \lor \psi \mid \neg \varphi \mid
\forall l : x \in a. \varphi(x, l) \mid \exists l : x \in a. \varphi(x, l)$

The intuitive meaning is just well-known set theoretical one.

Note that only bounded quantifiers are allowed.

Since all (hyper)sets are assumed to be finite, this makes the values of $\triangle$-terms and $\triangle$-formulas computable, and even in polynomial time.

Moreover, a natural extension of the above syntax by a recursion (Rec) and hyperset decoration (Dec) operators captures exactly PTIME over (finite) hypersets [Sazonov 1987,93,94], [Lisitsa,Sazonov 1997].

Another version of $\triangle$ captures exactly LOGSPACE over (finite) well-founded sets [Lisitsa,Sazonov 1997], [Leontjev,Sazonov 2001].
Recursion operator

Recursive \( \Delta \)-Separation is straightforward extension from the framework of first order logic to \( \Delta \):

the following is considered as new \( \Delta \)-term construct

\[
\text{Rec } p. [p = \{l : x \in a \mid \varphi(l, x, p)\}]
\]

for \( \varphi \) any \( \Delta \)-formula (depending on set variable \( p \) positively).

Let \( p_0 \overset{\text{def}}{=} \emptyset \) and \( p_{i+1} \overset{\text{def}}{=} \{l : x \in a \mid \varphi(l, x, p_i)\} \).

Then this \textit{monotonic} sequence of subsets of \( a \) must \textit{stabilise}, and the result is considered as the \textit{value of the recursion operator}. 
Decoration operator

For any hyperset $g$, some its elements may happen to be ordered pairs

$$\langle u, v \rangle \overset{\text{def}}{=} \{\text{Fst} : u, \text{Snd} : v\}.$$  

In general, these are labelled ordered pairs $l : \langle u, v \rangle$ in $g$ considered also as labelled edges $u \xrightarrow{l} v$ of a graph represented in this way by the hyperset $g$.

All other non-pair elements of $g$ may be ignored.

Thus,

- any (hyper)set $g$ can be considered as a graph,
- and any other set $v$ can be considered as its vertex (possibly isolated, if $v$ does not participate in the above ordered pairs).
Decoration operator

\[ \text{Dec}(g, v) = \text{Dec}_g(v) \]

is the *unique hyperset* represented by the vertex \( v \) of \( g \) according to AFA for graphs.

Intuitively, \( \text{Dec} \) is also a *plan performance operator*:

\( v \)-rooted graph \((g, v)\) serves as a *plan of building of a new hyperset* (or complex data) \( \text{Dec}(g, v) \), according to this plan.
Decoration operator

Example 1.

\[ g = \{ \langle u, v \rangle, \langle v, w \rangle, \langle u, w \rangle \} \]

represents the (acyclic) graph

\[ u \to v, \ v \to w, \ u \to w. \]

Then \( \text{Dec}_g(w) = \emptyset \), \( \text{Dec}_g(v) = \{\emptyset\} \), and \( \text{Dec}_g(u) = \{\{\emptyset\}, \emptyset\} \).

Example 2. For the loop-graph \( \bigcirc = \{ v \to v \} \),

\[ \text{Dec}_\bigcirc(v) = \Omega \]

is (according to AFA) a unique hyperset which satisfies, the equality

\[ \Omega = \{\Omega\} \]

This “cyclic” hyperset is not definable in \( \Delta \) without \( \text{Dec} \).
Decoration operator

These examples are very simple, but rather abstract.

In more practical flavour:

Graph $g$ can describe a plan of constructing a system of Web or WDB pages.

Then the action of Dec would be something like a transformation of this plan to a real system of WDB pages hyperlinked as required in this plan.

Thus, Dec is a quite reasonable WDB construct.
Operational semantics of $\Delta$-queries

Given any WDB as a system of set equations $\bar{s} = \bar{B}(\bar{s})$, we want to compute

- **hyperset value** of any $\Delta$-term $t$ or
- **Boolean value** of any $\Delta$-formula $\varphi$,

assuming that they involve only set names (constants) $\bar{s}$ from this WDB and no free variables.

The process of **computation** consists in a sequence of reduction steps

$$ WDB_0 \triangleright WDB_1 \triangleright WDB_2 \triangleright \ldots \triangleright WDB_N = WDB'. $$

$WDB_0$ is the initial WDB **extended by non-flat equation** $res = t$.

The last $WDB' = WDB_N$ consists only of flat set equations, including the result of **“flattening”** the equation $res = t$.

The unique **abstract hyperset** denoted by $res$ in $WDB' = WDB_N$ is considered as the **computed value** of $t$.

In the case if the query is a $\Delta$-formula $\varphi$, we include in $WDB_0$ the **Boolean equation** $res = \varphi$ with the intention to get a **“flattened”** value $\text{true}$ or $\text{false}$. 
Operational semantics of $\triangle$-queries
(the pure case)

Let us assume that $s, p, \ldots, q$, participating in the following below reduction rules, be some selected set names with corresponding (recursive) set equations from the initial WDB:

$$s = \{s_1, \ldots, s_m\},$$
$$p = \{p_1, \ldots, p_n\},$$
$$\ldots$$
$$q = \{q_1, \ldots, q_k\}.$$

Names $s_i, p_j, \ldots, q_l$ may coincide one with another, or with some of $s, p, \ldots, q$.

Reduction Rules:

Here $res$ is just the left-hand-side of the equation to be reduced.

$$res = t(t_1, \ldots, t_a) \triangleright \begin{cases} res & = t(res_1, \ldots, res_a), \\ res_1 & = t_1, \\ \cdots \\ res_a & = t_a, \\ res = \{s, p, \ldots, q\} & \text{— no reduction is required for set names } s, p, \ldots, q \\ res = s \cup p \cup \ldots \cup q & \triangleright res = \{s_1, \ldots, s_m, p_1, \ldots, p_n, \ldots, q_1, \ldots, q_k\} \end{cases}$$
Reduction Rules

\[ \text{res} = \bigcup s \triangleright \text{res} = s_1 \cup \ldots \cup s_m, \]
\[ \text{res} = \{ t(x) \mid x \in p \} \triangleright \text{res} = \{ t(p_1), \ldots, t(p_n) \}, \]
\[ \text{res} = \{ x \in p \mid \varphi(x) \} \triangleright \text{res} = \{ p_{i_1}, \ldots, p_{i_n} \} \text{ with } p_{i_j} \text{ all those set names } p_i \]
  for which \( \text{res}_i = \varphi(p_i) \triangleright \text{res}_i = \text{true} \)
  (and \( \text{res}_i = \varphi(p_i) \triangleright \text{res}_i = \text{false} \) for all other \( i \)),
\[ \text{res} = \forall x \in p \varphi(x) \triangleright \text{res} = \varphi(p_1) \& \ldots \& \varphi(p_n) \ (= \text{true} \text{ if } n = 0), \]
\[ \text{res} = \text{true} \& \text{true} \triangleright \text{res} = \text{true}, \]
\[ \text{res} = \varphi \& \text{false} \triangleright \text{res} = \text{false}, \]
\[ \text{res} = \text{false} \& \varphi \triangleright \text{res} = \text{false}, \]
\[ \text{res} = \neg \text{true} \triangleright \text{res} = \text{false}, \]
\[ \text{res} = \neg \text{false} \triangleright \text{res} = \text{true}, \]
\[ \text{res} = s \in p \triangleright \text{res} = \exists x \in p (s = p), \]
\[ \text{res} = (s = p) \triangleright \ldots \] (bisimulation relation has been considered above).

We omit the rules for transitive closure, recursion, and decoration, operators \( \text{TC}, \text{Rec}, \) and \( \text{Dec}. \)
Queries preserve equality
All $\Delta$-queries over hypersets definable in terms of systems of set equations and the above operational semantics are in *complete accordance* with the bisimulation equivalence relation:

- they are *bisimulation invariant*
- that is, *preserve the equality relation between abstract hypersets*:

$$\bar{x} = \bar{y} \implies q(\bar{x}) = q(\bar{y}), \quad \text{if } q \text{ is a } \Delta\text{-term},$$
$$\bar{x} = \bar{y} \implies q(\bar{x}) \Leftrightarrow q(\bar{y}), \quad \text{if } q \text{ is a } \Delta\text{-formula}$$

where $\bar{x}$ and $\bar{y}$ are set names. For example,

$$s_i = s_j \implies \bigcup s_i = \bigcup s_j.$$ 

This also means that $=$ is a *congruence* relation.

It is by this reason that we denote bisimulation just as $=.$

This represents more abstract set theoretic level of thought —not a graph theoretic one—although closely related.
Path expressions

In general, define path expression either as

- $\Lambda$, the empty path expression, or
- a label constant, or
- a label variable, or
- $\_\,$, underscore (meaning “any label”), or
- $p \cdot q$, concatenation of path expressions, or
- $q \langle z \rangle p$, concatenation by a set variable or a set name $z$, or
- $(p)^*$, iteration, or
- $(p)^?$, optional path expression, or
- $p | q$, alternation of path expressions (either $p$ or $q$), or
- $p || q$, parallel composition or branching (both $p$ and $q$).

Path expressions defined above extend those in UnQL called there patterns.
Path expressions

Before including path expressions in $\Delta$, let us give formal inductive definitions of two auxiliary kinds of expressions

- **the assertion** $p\langle t \rangle$: “there exists, possibly $||$-branching path $p$ from $t$ via all vertices and labels mentioned in $p$”, and

- **the set** $[p\langle t \rangle] \subseteq TC(\{t\})$: “the set of all $x$ such that there exists a path $p$ from $t$ via all vertices and labels mentioned in $p$ all whose branches converge to the same $x$ (denoted also as $\langle x \rangle p\langle t \rangle$)”.

In fact, $p\langle t \rangle$ and $[p\langle t \rangle]$ are a $\Delta$-formula and $\Delta$-term, respectively, depending on set and label variables occurring in $p$.

The above should be understood over any non-redundant (strongly extensional) WDB-graph.
Defining $p\langle t \rangle$ and $[p\langle t \rangle]$ in $\triangle$

\[
\langle t \rangle \overset{\text{def}}{=} \text{true}, \\
_\langle t \rangle \overset{\text{def}}{=} t \neq \emptyset, \\
\langle t \rangle \overset{\text{def}}{=} \exists m : x \in t (m = l), \\
q.p\langle t \rangle \overset{\text{def}}{=} \exists y \in [p\langle t \rangle](q\langle y \rangle), \\
q^z p\langle t \rangle \overset{\text{def}}{=} z \in [p\langle t \rangle] \& q\langle z \rangle, \\
(q)^*\langle t \rangle \overset{\text{def}}{=} \text{true}, \\
(p)\?\langle t \rangle \overset{\text{def}}{=} \text{true}, \\
(p|q|\cdots|r)\langle t \rangle \overset{\text{def}}{=} p\langle t \rangle \lor q\langle t \rangle \lor \cdots \lor r\langle t \rangle, \\
(p||q||\cdots||r)\langle t \rangle \overset{\text{def}}{=} p\langle t \rangle \& q\langle t \rangle \& \cdots \& r\langle t \rangle;
\]
Defining $p(t)$ and $[p(t)]$ in $\Delta$

$[[t]] \overset{\text{def}}{=} \{ t \}$,
$[-(t)] \overset{\text{def}}{=} \{ x \mid m : x \in t \}$,
$[l(t)] \overset{\text{def}}{=} \{ x \mid m : x \in t \& m = l \}$,
$[q.p(t)] \overset{\text{def}}{=} \bigcup \{ [q(y)] \mid y \in [p(t)] \}$,
$[q(z)p(t)] \overset{\text{def}}{=} [q(z)] \text{ if } z \in [p(t)]$,

$[(q)^{\ast}(t)] \overset{\text{Rec}}{=} \{ t \} \cup \{ x \mid \exists y \in [q(t)] (x \in [((q)^{\ast}y)]) \}$
$\overset{\text{Rec}}{=} \{ x \mid x = t \lor \exists y \in [q(t)] (x \in [((q)^{\ast}y)]) \}$
$\overset{\text{Rec}}{=} \{ x \in TC(\{ t \}) \mid x = t \lor \exists y \in [q(t)] (x \in [((q)^{\ast}y)]) \}$,

$[(p)^{?}(t)] \overset{\text{def}}{=} [p(t)] \cup \{ t \}$,

$[(p|q|\cdots|r)(t)] \overset{\text{def}}{=} [p(t)] \cup [q(t)] \cup \cdots \cup [r(t)]$,

$[(p||q||\cdots||r)(t)] \overset{\text{def}}{=} [p(t)] \cap [q(t)] \cap \cdots \cap [r(t)]$. 
Defining $p\langle t \rangle$ and $[p\langle t \rangle]$ in $\triangle$

In particular,

$$\langle x \rangle l \langle t \rangle \iff l : x \in t,$$
$$\langle z \rangle p \langle t \rangle \iff z \in [p\langle t \rangle],$$
$$[\langle z \rangle p \langle t \rangle] = \{z\} \text{ if } z \in [p\langle t \rangle],$$
$$q\langle z \rangle p \langle t \rangle \iff q\langle z \rangle \& \langle z \rangle p \langle t \rangle,$$
$$p(\bar{z}) \langle t \rangle \Rightarrow \bar{z} \in TC(\{t\}).$$

The last implication is the formal confirmation of the idea that path expressions are a special case of the transitive closure operator $TC$.

This finishes the definition of semantics of the expressions $p\langle t \rangle$ and $[p\langle t \rangle]$. 
Defining semantics of $\Delta$-language extended by path expressions

Now, we extend $\Delta$, without changing its expressive power, by new kind of terms and formulas involving arbitrary path expressions.

In particular, we generalise bounded quantification.

This extension is only syntactic sugaring of $\Delta$ because the new meaning is definable in the original version of the language:

$$\exists p(\bar{z})\langle t \rangle \varphi(\bar{z}) \overset{\text{def}}{=} \exists \bar{z} \in TC(\{t\})(p(\bar{z})\langle t \rangle \& \varphi(\bar{z})),$$

$$\{ s(\bar{z}) \mid p(\bar{z})\langle t \rangle \& \varphi(\bar{z}) \} \overset{\text{def}}{=} \{ s(\bar{z}) \mid \bar{z} \in TC(\{t\}) \mid p(\bar{z})\langle t \rangle \& \varphi(\bar{z}) \} ,$$

$$\bigcup_{p(\bar{z})\langle t \rangle} S(\bar{z}) \overset{\text{def}}{=} \bigcup \{ S(\bar{z}) \mid p(\bar{z})\langle t \rangle \} .$$

All (set and label) variables $\bar{z}$ in a path expression $p(\bar{z})$ are quantified above.
Conceptual comparison of $\Delta$
with UnQL and UnCAL

Query languages UnQL and UnCAL where introduced and developed in [Buneman, Davidson, Hillebrand, and Suciu, 1996] and [Buneman, Fernandez, and Suciu, 2000]:

— most analogous to $\Delta$ with considering a database as a graph with labelled edges, up to bisimulation relation,

— however, because of “input” and “output” vertices, the data, considered in UnQL and UnCAL are not exactly hypersets, although can be imitated by hypersets in $\Delta$,

— mostly treated as graphs in UnQL and UnCAL, and the imitation in $\Delta$ goes, anyway, also in terms of graphs.

So, UnQL and UnCAL are essentially graph languages with

• more stress on the operational/computational meaning and

• no explicit mentioning hypersets as a fundamental concept.
Conceptual comparison of $\Delta$
with UnQL and UnCAL

In contrast, $\Delta$ *almost completely can be understood in terms of sets* and is very intuitive essentially by the same reason why set theory proved to be the most clear and widely accepted during the last century approach giving the foundation of mathematics.

On the other hand, the *operational semantics* of $\Delta$ is also essentially *based on graphs* or, equivalently, on textual *systems of set equations*.

Elementary set theoretic concepts are, in fact, *widely known*. 

XML *representation of (hyper)sets* or systems of set equations can be also used very naturally.
Technical comparison of $\Delta$
with UnQL and UnCAL

Graphs from hypersets

Each hyperset $x$ defines in $\Delta$ (and is itself defined by) a graph

$$G(x) = \langle V(x), E(x) \rangle$$

$$V(x) \overset{\text{def}}{=} TC(\{x\})$$

$$E(x) \overset{\text{def}}{=} \{ l : \langle u, v \rangle \mid u, v \in V(x) \& l : v \in u \}$$

with $l : \langle u, v \rangle$ written also as the graph edge $u \xrightarrow{l} v$.

Then $x$ is the natural root and serves as an input vertex of the graph $G(x)$, and evidently

$$\text{Dec}(G(x), x) = x.$$
Technical comparison of $\Delta$
with UnQL and UnCAL

Append operation of UnCAL—simplest version

Define in $\Delta$ a simple version of the *append* operation for two hypersets

$$u @_l v \overset{\text{def}}{=} \text{Dec}(G^l_{u,v}, u)$$

where the graph $G^l_{u,v}$ is the result of “glueing” together of three vertices \(\{l : \emptyset\}, \emptyset \in V(u)\), and \(v \in V(v)\) in graphs $G(u)$ and $G(v)$.

Here \(l\) serves to label an *output* in \(u\) to be connected with the natural root of \(v\) playing the role of the *input* in \(v\).

That is, \(v\) is “*substituted*” in \(u\) in place of \(\{l : \emptyset\}\).

Example, \(\{k : \{l : \emptyset\}, m : z\} @_l v = \{k : v, m : z\}\).
Technical comparison of $\Delta$ with UnQL and UnCAL

Append operation of UnCAL—more general versions (some also going outside of UnCAL) for the case of

- appending *not only to leafs*
- *multiple inputs and outputs*
- *multiple hypersets* $u_i$ (graphs $G(u_i)$ connected as a *chain* $G = \{u_0 \rightarrow u_1 \rightarrow \cdots \rightarrow u_n\}$ (of indefinite finite length) of hypersets as

$$\hat{\oplus}(G) \overset{\text{def}}{=} u_0 \hat{\oplus} u_1 \hat{\oplus} \cdots \hat{\oplus} u_n$$

or even as *arbitrary graph* $G$.

In each case this is *definable* in $\Delta$, and the participation of *decoration Dec is, in fact, crucial* (as in the previous slide).
Technical comparison of $\Delta$ with UnQL and UnCAL: Structural recursion

In the case of well-founded sets (with no cycles) the general set theoretic vertical (that is going deeply to elements of $x$, etc.) structural recursion schema is defined by

$$f(x) = \bigcup_{l:y \in x} e(l, y, f(y))$$

A similar recursion schema over sets

$$f(x) = g(\bigcup_{u \in x} f(x), x)$$

was introduced earlier by [Jensen and Karp, 1967]. Both are set theoretic analogues of primitive recursion and therefore cannot be computable in polynomial time in general.
Technical comparison of $\Delta$
with UnQL and UnCAL:
Structural recursion

Although it is problematic to define general structural recursion for arbitrary (possibly cyclic) hypersets, this can be done, and even in $\Delta$, for its restricted version used in UnCAL:

$$
f(x) = \bigcup_{l:y \in x} e(l, y) @ f(y)
$$

The above general form of append $@$ is used here, as well as for even stronger horizontal version of structural recursion:

$$
f(x, u) = \bigcup_{l:(u, v) \in x} e(x, l, v) @ f(x, v)
$$

which has in $\Delta$ the power equivalent to Dec.

This recursion also looks at the structure of $x$, but only at a superficial, near to the root of $x$ “horizontal” level.
**Expressive power of $\Delta$ vs. UnQL and UnCAL**

By using the above and following below considerations we conclude that

The *expressive power* of $\Delta$ (which is exactly PTIME) is strictly *stronger* than UnQL and UnCAL.

**Conjecture.** It seems plausible that the general form of the *decoration* operations $\text{Dec}$ is *undefinable* even in $\Delta - \text{Dec} + \text{UnCAL}$ and that the *horizontal structural recursion is strictly stronger that the vertical one*.

The precise expressive power of general queries in UnCAL and in UnQL remains *unclear*.

However, it is known that *all UnCAL queries can be expressed in FO + TC*.

Note that UnQL and UnCAL *even do not include the equality* (bisimulation) predicate for the data—for the sake of efficiency of query evaluation.

However, the *equality/bisimulation* of data is, anyway, *necessary to use in the semantics of path expressions* (participating in UnQL).

As general equality is also not included in UnCAL, it is also unclear that UnQL, involving equality implicitly (in the semantics of path expressions), can be really interpreted in UnCAL, as it is assumed in the paper [P. Buneman, M. Fernandez, and D. Suciu, 2000].
Expressive power of $\Delta$ vs. UnQL and $\text{UnCAL}$

There is, however, Theorem 3 in the op. cit. implying that the expressive power of $\text{UnCAL}$ restricted to the ordinary relational databases (i.e., acyclic data graphs/trees of a very special form and the depth three) is exactly that of $\text{FO}$, or of the relational calculus. Here even the full power of $\text{UnCAL}$ is used allowing construction of arbitrary data graphs in the intermediate steps of query evaluation.

But for the case of $\Delta$ we should get here exactly $\text{FO} + \text{LFP}$, as well as for arbitrary hypersets (or graphs representing them). That is, $\Delta$ has a stronger power which, moreover, has a precise characterisation—$\text{FO} + \text{LFP}$, or $\text{PTIME}$, if there is a linear order on labels (or if there are no labels at all).

There are also versions of $\Delta$ exactly corresponding to $\text{FO} + \text{TC}$, or $\text{FO} + \text{DTC}$ (in fact, $\text{LOGSPACE}$), but only in the case of acyclic sets/graphs.

It would be very interesting to extend these results on $\text{LOGSPACE}$ to the case of arbitrary (cyclic) hypersets.