Development of Unit Calculation Algebra as an Application Function of the Cellular Data System

Toshio Kodama¹, Tosiyasu L. Kunii², Yoichi Seki³
¹Venture Business Department, Maeda Corporation, 2-10-26 Fujimi, Chiyoda-ku, Tokyo 102-0071 Japan kodama.ts@jcity.maeda.co.jp
²IT Institute, Kanazawa Institute of Technology, 1-15-13 Jingumae, Shibuya-ku Tokyo 150-0001 Japan tosi@kunii.com; http://www.kunii.com/
³Software Consultant 3-8-2 Sakaemachi, Hino City, Tokyo 191-0001 Japan ysekiel@mail.hinocatv.ne.jp

Abstract

Cyberworlds are information worlds where data and its dependencies are always changing and may evolve unpredictably, just like living things. Thus far, it has been difficult to model cyberworlds with higher freedom using the existing model because of its complexity. We believe that Incrementally Modular Abstraction Hierarchy (IMAH), which has an abstraction hierarchy preserving invariants from the abstract to the concrete level, and which was invented by one of the authors (T. L. Kunii), makes it possible. We have developed a data processing system called Cellular Data System (CDS) based on IMAH. With CDS, you can express any space, with forms or not, and attach or divide spaces by an equivalence relation to output information like you do in your head. In this paper we have designed and implemented wide-use algebra called unit calculation algebra on the presentation level in IMAH as an application function of CDS, and we have shown its effectiveness through examples.

Keywords: cyberworlds, incrementally modular abstraction hierarchy, formula expression, topological space, adjunction space, cellular space, presentation level

1. Introduction

Cyberworlds, where commercial transactions, product design, and material supply take place all over the world through the Internet, are bringing about combinatorial explosion because data dependency is constantly changing. Cyberworlds are more complicated and fluid than any other previous worlds in human history, and are constantly evolving. The number of companies that conduct business in cyberspace, such as Google and eBay, is increasing and the market is growing remarkably.

On the other hand, in the development of business application systems in each company, the frequent failure of the customer side and the supplier side to agree causes increases in development costs and delays in development time. The same is true for a developed system. Due to maintenance difficulties, current systems cannot meet changes in user requirements, so trouble frequently arises from mismatches in business specifications and system functions. In some cases, a huge system as the mainstay system in a company, where the number of program steps is hundreds of million, needs several years to develop and increases in development cost squeeze management.

Is there a model that can reflect both completely different worlds, cyberworlds which are ever-evolving with higher freedom, and real worlds, which are built like business application systems?

Incrementally Modular Abstraction Hierarchy that one of authors (T. L. Kunii) proposes can model the architecture and the changes of cyberworlds from a general level (the homotopy level) to a specific one (the presentation level, the view level), preserving invariants while preventing combinatorial explosion. It also benefits the reuse of information, guaranteeing modularity of information based on the mechanism of disjoint union. Unlike IMAH, other leading data models, like the relational data model, the object oriented data model, the entity-relation model, and XML do not support the disjoint union or the attaching function by equivalence relation to guarantee and validate invariant preservation.

In this research, one of authors (Y. Seki) proposed an algebraic system called Formula Expression as a business application development tool to realize IMAH. T. Kodama has actually implemented the system using Formula Expression. We named the system the Cellular Data System. In this paper, we put emphasis on practical
use by taking up some examples. First, we have designed and implemented a useful new algebra called unit calculation algebra on the presentation level in the IMAH to calculate data in business work. We have demonstrated the effectiveness of CDS by developing a general business application system (a customer information management system) and abbreviating the process of implementing most application programs.

In Chapter 2, we mention IMAH and Formula Expression briefly. We design the unit calculation algebra and its function in Chapter 3. In Chapters 4 and 5, we explain the examples of the two business applications. In Chapter 6, we survey related works, and we summarize our conclusions in Chapter 7.

2. Incrementally Modular Abstraction Hierarchy (IMAH) and Formula Expression

IMAH

The following list is the Incrementally Modular Abstraction Hierarchy to be used for defining the architecture of cyberworlds and their modeling:

1. the homotopy (including fiber bundles) level
2. the set theoretical level
3. the topological space level
4. the adjunction space level
5. the cellular space level
6. the presentation (including geometry) level
7. the view (also called projection) level

In modeling cyberworlds in cyberspaces, we define general properties of cyberworlds at the higher level and add more specific properties step by step while climbing down the incrementally modular abstraction hierarchy. A cyberspace is specified by the product of a base space and a bundle of fibers called a fiber bundle. The product space constitutes cyberspaces. The properties defined at the homotopy level are invariants of continuous changes of functions. Homotopy is a Greek origin terminology to signify continuous deformation in a general sense. The properties that do not change by continuous modifications in time and space are expressed at this level. At the set theoretical level, the elements of a cyberspace are defined, and a collection of elements constitutes a set with logical operations. When we define a function in a cyberspace, we need domains that guarantee continuity such that the neighbors are mapped to a nearby place. Therefore, a topology is introduced into a cyberspace through the concept of neighborhood. Cyberworlds are dynamic. Sometimes cyberspaces are attached together, an exclusive union of two cyberspaces, where attached areas of two cyberspaces are equivalent. It may happen that an attached space is obtained. These attached spaces can be regarded as a set of equivalent spaces called a quotient space that is another invariant. The example of an online shop at this level is shown in Fig 2.1. At the cellular structured level, an inductive dimension is introduced into each cyberspace. At the presentation level, each space is represented in a form which may be imagined before designing cyberworlds. At the view level, the cyberworlds are projected onto view screens. This level has been well studied and developed as computer graphics, a solid academic discipline.

The definition of Formula Expression

Formula Expression in the alphabet is the result of finite times application of the following (1)-(7).

(1) a \( (a \in \Sigma) \) is Formula Expression
(2) unit element \( \varepsilon \) is Formula Expression
(3) zero element \( \varnothing \) is Formula Expression
(4) when \( r \) and \( s \) are Formula Expression, addition of \( r + s \) is also Formula Expression
(5) when \( r \) and \( s \) are Formula Expression, multiplication of \( r \times s \) is also Formula Expression
(6) when \( r \) is Formula Expression, \( (r) \) is also Formula Expression
(7) when \( r \) is Formula Expression, \( \{r\} \) is also Formula Expression

Strength of combination is the order of (4) and (5). If there is no confusion, \( \times \), \( (\) \), \( \{\} \) can be abbreviated. + means disjoint union and is expressed as \( \sqcup \) specifically and \( \times \) is also expressed as \( \times \). In short, you can say "a formula consists of an addition of terms, a term consists of a multiplication of factors, and if the \( () \) or \{\} bracket is added to a formula, it becomes recursively the factor". In Formula Expression, four maps (the expansion map, the
bind map, the division map, the attachment map) are defined. [19]

**The grammar that generates Formula Expression**

The grammar that generates Formula Expression is defined as follows:

You assume that $\Sigma_L$ is a set of ideograms and its element is $w \ (\in \Sigma_L)$.

$$G = (\{E, T, F, id\}, \Sigma_L \cup \{+, \times, (, ), \{, \}, \}, P, E),$$

$$P = \{E \rightarrow T|E+T, T \rightarrow F|T\times F, F \rightarrow (E)|\{E\}|id, id \rightarrow w\}$$

Here, $E$ is called a formula, and $T$ is called a term, and $F$ is called a factor, and $id$ is called an identifier; $+$ is called an addition operation, $\times$ is called a multiplication operation, $( )$ is called a 1st bracket, and $\{ \}$ is called a 2nd bracket. If you paraphrase the above-mentioned grammar, you can say "a formula consists of an addition of terms, a term consists of a multiplication of factors, and if the 1st or 2nd bracket is added to a formula, it becomes recursively the factor". And when a term is a component of a formula, we say that the formula has the term. And when a factor of the bracket that includes a term is a component of a formula, we say that the formula includes the term. It is the same with a term and a factor. An example is shown below.

The term "a(b+c){d+e}" has factors "a", "(b+c)", "{d+e}", and includes factors "b", "c", "d", "e".

**The design of spaces in IMAH**

We designed the spaces on each level in IMAH by using Formula Expression. In this paragraph, we mention a topological space, an adjunction space and a cellular space.

A topological space is a space that can express an element as a subset. Hence, the formula for a subset is as follows:

$$\Pi(\{\,\text{an element}\,\}| \text{an element}) \text{ or } |\{\,\text{an element}\,\}|$$

And the formula for a topological space is:

$$id^{\times}(\{\,\text{a subset}\,\})$$

"id" is the factor that identifies an element and an element is always a term. If there is no confusion, the bracket () can be abbreviated.

A quotient space is created by the division of a topological space according to equivalence relations. An adjunction space is created by attaching according to equivalence relations between quotient spaces. Hence, the formula for a topological space changes to the formula for a quotient space via the division map by the factor that expresses equivalence relations, and the quotient terms that have each factor expressing equivalence relations change to the formula for an adjunction space by attaching via the attachment map. Let $X$ and $Y$ be the formulas of topological spaces. If a factor $x$ that is included in $X$ and a factor $y$ that is included in $Y$ are equivalent, namely $x \sim y$, the formulas of quotient spaces are created via division map by the factors $x$ and $y$. If you assume each quotient term of $X$, $Y$ to be $r^{\times}s$, $t^{\times}u$, the formula for quotient spaces is as follows:

$$a \text{ remainder term of } X + r^{\times}s$$
$$a \text{ remainder term of } Y + t^{\times}u$$

Next, the formula for the adjunction space is created by attaching quotient terms of $X$ and $Y$ via attachment map by $x$ and $y$.

$$a \text{ remainder term of } X$$
$$+ a \text{ remainder term of } Y$$
$$+ \{r+s\}{x+y}\{t+u\}$$

A cellular space is a space that can express dimensions. In this case, the dimensions correspond with the attributes of a cellular space. Hence, the formula for a cellular space is as follows:

$$id^{\times} \text{ a primary key attribute}\times \{\varepsilon + i\,\text{a subset}\,\} \times (\{\,\text{an instance}\,\}\times \{\varepsilon + |\,\text{a subset}\,\}\times \{\,\text{expresses values}\,\}))$$

"id" is the factor that identifies a cellular space and "an instance id" is the factor that identifies an instance.

3. **Unit calculation algebra on the presentation level**

**The grammar that generates unit calculation algebra**

The presentation level is the level where objects have the concept of types. We define algebra called unit calculation algebra on the presentation level. We define the grammar that generates it below. You assume that a set of real numbers is $R$ and its element is $p \ (\in R)$, and that a set of ideograms to be $L$ and its element is $u \ (\in L)$.

$$G = (\{E, T, F\}, \Sigma \cup \{*, /\}, P, E)$$

$$P = \{E \rightarrow T|E\times T, T \rightarrow F|T\times F, F \rightarrow (E)|\{E\}|id, id \rightarrow p\}$$
Here, E is called a unit formula, T is called a unit term, F is called a unit factor, p is called a numerical factor, and u is called a letter factor. As an example, a unit formula "€5/piece*10pieces + €10/piece*20pieces" is created in Fig 3.1. If there is no confusion, each operation sign can be abbreviated. And is sometimes expressed as Σ. This algebra is one in which some rules are added to the id of Formula Expression, and the id of Formula Expression corresponds to a unit formula E in the unit calculation algebra.

The properties of unit calculation algebra

If we assume that p, q, r, s, m, n are arbitrary numerical factors, and that u, v, w, y are arbitrary letter factors, the unit calculation algebra has the following properties.

1. \( u_0 = \varepsilon \)
2. \( u^1 = u \)
3. \( 1u = u \)
4. \( 0u = 0 \)
5. \( u^m u^n = u^{m+n} \)
6. \( (u^m)^n = u^{mn} \)
7. \( (uv)^n = u^n v^n \)
8. \( p(uv) = (pu)(qv) \)
9. \( p(\sum) = \sum(pu) \)
10. \( pu = qv \)
11. \( pu = u^p \)
12. \( pu = pu^p \)
13. \( pu = pu^p qv \)
14. \( pu = pu^p qv + pu^p r \)
15. \( pu = pu^p qv + pu^p r + pu^p qv + pu^p r \)
16. \( pu = pu^p qv + pu^p r + pu^p qv + pu^p r \)
17. \( pu = pu^p qv + pu^p r + pu^p qv + pu^p r \)
18. \( pu = pu^p qv + pu^p r + pu^p qv + pu^p r \)
19. \( pu = pu^p qv + pu^p r + pu^p qv + pu^p r \)
20. \( pu = pu^p qv + pu^p r + pu^p qv + pu^p r \)
21. \( pu = pu^p qv + pu^p r + pu^p qv + pu^p r \)
22. \( pu = pu^p qv + pu^p r + pu^p qv + pu^p r \)
23. \( pu = pu^p qv + pu^p r + pu^p qv + pu^p r \)

If you want express "the total of €10 and $10 in ¥ when €1 is ¥163 and $1 is ¥122", it is expressed and calculated as follows on the presentation level.

\[
\begin{align*}
\text{€10} \times \text{¥163} / \text{€1} + \text{€10} \times \text{¥122} / \text{$1} &= \text{¥1,630} + \text{¥1,220} \\
&= \text{¥2,850}
\end{align*}
\]

A unit calculation map \( f \) and the transfer to the presentation level map \( g \)

A unit calculation map \( f \) is defined based on the above mentioned properties. If you assume the entire set of the unit calculation algebra to be \( A \), \( f: A \rightarrow A \).

We assume that the map that repeats the map \( f \) until pre-image and image by \( f \) are the same is \( f_w \). A simple example of calculating accommodation costs is shown below.

\[
\begin{align*}
\text{f}_w((\text{€100}/(1\text{person*1day}) \times 15\text{persons} + \text{€120}/(1\text{person*1day}) \times 10\text{persons}) / \text{room*10rooms*3days}) &= \text{€81,000}
\end{align*}
\]

In this example, we assume that a singular form and a plural form of letter factors such as "day" and "days" are equal. We defined the map \( g \) that changes + to * and × to * in formulas, and it is called the transferring to the presentation level map.

Implementation

This system is a web application developed using JSP and Tomcat 5.0 as a Web server. The client and the server are the same machine. (OS: Windows XP; CPU: Intel Pentium 3, 1.2GHz; RAM: 1.1Gbyte; HD: 20GB)

Fig 3.4 is the flowchart of the algorithm for the
calculation of a unit formula. This expresses the main flow of the process; the details are abbreviated. Supplementary explanations follow:

**Fig 3.4 The flowchart of the algorithm for the calculation of a unit formula**

- The focus is the recursive process that is done if a coming unit factor is of the type (). It is possible because the content of the unit factor of the type () is a unit formula recursively extending topological processing.
- In the calculation for a unit term, each unit factor in the term is separated into a numerical factor and a letter factor, and they are calculated separately. They are then attached to become a unit factor again and are transferred to the next process.
- In the case of an addition or a subtraction operation of unit factors, if the letter factors are not the same, the process doesn’t continue. A simple example is that letter factors ¥ and € in ¥50 €5 are different, so this is not calculated further.

4. **Case study: A customer information**

**management system**

**Data input and output**

In a business application system to manage customer information, if customer data can be inputted into and outputted from the system flexibly, and if you can get the related information from other systems, the system functions can be considered to be greatly improved. CDS does this. In this chapter, we take up some examples where data is simplified without losing generality. We have also implemented a simple user interface.

Data input as a cellular space (2.3) possesses the following flexibilities and greatly improves system maintainability:

1. There is no need for file design. Both schema data and instance data are expressed as formulas, and the mechanism of disjoint union of spaces is supported by addition operation + in CDS. Therefore, there is no need for file design because you can design schema for each instance.
2. It is possible to detail values. Multiplication operation × is supported in CDS, so it is possible for the application system to meet the requirements of detailing values.
3. It is possible to input plural values. Recursive expression by (), {} is supported in CDS, so you can input plural values into the application system.

We take up the example of inputting customer data as a cellular space.

formula 4-1:

```
customerInformation×customerCode{ε+name+company +age+univ+hobby}{T1{ε+Kojima+T-corp.×marketingDepartment+30+W-University+{golf+chess}}+customerInformation×customerCode{ε+name+company+age+univ+hobby+character}{F1{ε+Nomi+Fcorp.+45+{K-University+T-University×masterDegree}×swiming+diligent}}+customerInformation×customerCode{ε+name+company+age +univ+hobby+family}{P1{ε+Akahoshi+P-corp.+36+F-University+golf+{wife+children×two}}}
```

Fig 4.11 is an example input from a user interface. Users can design files, and can input detailed values and plural values.

Next, the following memo data about a customer is inputted. Then, the formulas for the topological spaces are created and inputted, and added to the above formula.

**Memo1:**

"On April 2, 2007 and May 5, 2007, Mr. Kodama of M-corp. met Mr. Kojima (divisional manager) and had a meeting about commercializing CDS. Mr. Kojima's
daughter graduated from J-University and became a
doctor. Mr. Kojima graduated from the same high
school as Mr. Yamada of S-corp. and is from Kunii
Laboratory of H-University.

If you want to obtain data connected to a specified word,
first you expand the inputted formula by the expansion
map, and second you get the bind term, if you bind the
result via the bind map assuming the specified word to
be the bind factor. For example, if you want the data on
"golf" from the inputted formulas, you can get the
following formula as a bind term from the customer data
and the product data if you bind the formulas by the bind
factor "golf".

\[
\text{formula 4-4:} \quad \{\text{customerInformation} \times \text{customerCode} \times \text{hobby}\{\text{T1}+\text{P1}\} + \text{productInformation} \times \text{productName}\{\text{S1}+\text{S2}+\text{S3}+\text{S4}\} \} \text{golf}\{\{e+c\}+\{\text{club1}+\text{club2}+\text{ball}+\text{shoes1}\}\}
\]

From this formula, you can know the fact "the hobby of
customers (customer codes are T1, P1) is golf and related
products are club, club2 and shoes1". Fig4.1-2 is an
example of an output to a user interface.

Fig 4.1-1 Data input from a user interface

Like this, data such as a memo without forms can be
inputted as the formula for a topological space.

Next, you create the formula for a cellular space for
customer data as data of another system and add it to the
above formula.

Like this, data such as a memo without forms can be
inputted as the formula for a topological space.

Next, you create the formula for a cellular space for
customer data as data of another system and add it to the
above formula.

If you want to know "the total price for the products that
Mr. Kojima bought", first you bind the inputted formula by
bind factors "T1", "purchase" and next if you arrange
the contents of the 3rd factor of the bind term by the 1st factors, you get the following formula as a bind term.

formula 4-6:
\[
\{\text{April} \times \text{Second} + \text{May} \times \text{Third}\} \times \text{Purchase}\left\{\{S1+S3+S4\}+\{S5\}\right\}\{\{¥20,000/\text{piece} \times 2\text{pieces}+¥100/\text{piece} \times 100\text{pieces}+¥18,000/\text{pair} \times 1\text{pair}\}+\{¥12,000/\text{pair} \times 1\text{pair}\}\}
\]

If you transfer the contents of the 4th factor of formula 4-6 to the presentation level by \(g\) and get an image of them by \(f_{gs}\), you can get the total price as follows.

\[
f_{gs}(¥(¥20,000/\text{piece} \times 2\text{pieces}+¥100/\text{piece} \times 100\text{pieces}+¥18,000/\text{pair} \times 1\text{pair}+¥12,000/\text{pair} \times 1\text{pair}))
\]

\[
=¥(¥20,000/\text{piece} \times 2\text{pieces}+¥100/\text{piece} \times 100\text{pieces}+¥18,000/\text{pair} \times 1\text{pair}+¥12,000/\text{pair} \times 1\text{pair})
\]

\[
=¥80,000
\]

If you want to know "the total price for the golf clubs sold (product code: S1)", first you divide the inputted formula by the factor "S1" and, next, if you bind the quotient terms by the factor "S1", you get the following formula as a bind term.

formula 4-7:
\[
\{\text{April} \times \text{Second} + \text{May} \times \text{Fifth}\} \times \text{Purchase}\left\{\text{S1}\right\}\{¥20,000/\text{piece} \times 2\text{pieces}+¥20,000/\text{piece} \times 1\text{piece}\}
\]

In the same way, if you get an image of the contents of the 4th factor of formula 4-7 by \(g\), \(f_{gs}\), you can get the total price as follows.

\[
f_{gs}(¥(¥20,000/\text{piece} \times 2\text{pieces}+¥20,000/\text{piece} \times 1\text{piece}))
\]

\[
=¥(¥20,000/\text{piece} \times 2\text{pieces}+¥20,000/\text{piece} \times 1\text{piece})
\]

\[
=¥60,000
\]

Considerations

First, if you use CDS according to the design of IMAH, you can input and output not only data with which schema is designed, but data with which schema couldn’t be defined before, such as memo data, as you can see in the above business application example. We also showed that you can input detailed values and plural values by the functions of \(\times\) (multiplication), + (addition) in CDS. We know that if you use the existing methods, when you want to detail values or deal with plural values, much maintenance work is needed because designed schema have to be modified. Next, the essential advance of the unit calculation algebra is that numerical values and unit information are expressed in the same way as factors of CDS and calculated respectively according to the defined four fundamental rules of arithmetic. So you don’t have to design unit information or its combination in the design process, separating data into a numerical value and unit information. You only have to input data that you recognize into the system. You will realize that there is no need for application program development or maintenance work in this system (except for the User Interfaces). This means that development costs and maintenance costs of the system can be cut dramatically.

5. Related works

The distinctive features of our research are the application of the concept of topological process, which deals with a subset as an element, and that the cellular space extends the topological space as you see in Chapter 2. Relational OWL as a method of data and schema representation is useful when representing the schema and data of a database. [6] But it is limited to representation of an object that has attributes. Our method can represent both objects: one that has attributes as a cellular space and one that doesn’t have them as a set or a topological space.

Many works applying other models to XML schema have been done. The motives of most of them are similar to ours. The approach in [11] aims at minimizing document revalidation in an XML schema evolution, based on a part of the graph theory. The X-Entity model [12] is an extension of the Entity Relationship (ER) model and converts XML schema to a schema of the ER model. In the approach of [9], the conceptual and logical levels are represented using a standard UML class and the XML represents the physical level. XUML [13] is a conceptual model for XML schema, based on the UML2 standard. This application research concerning XML schema is needed because there are differences in the expression capability of the data model between XML and other models. On the other hand, objects and their relations in XML schema and the above models can be expressed consistently by CDS, which is based on IMAH. That is because the tree structure, on which the XML model is based, and the graph structure, on which the UML and ER models are based, are special cases of a topological structure mathematically. Entity in the models can be expressed as the formula for a cellular space in CDS. Moreover, the relation between subsets, as we showed in 3.2, cannot in general be expressed by XML.

Although CDS is similar to the existing deductive database in expression, the two are completely different. The deductive database [17] raises the expression capability of the relational database (RDB) by defining some rules. On the other hand, CDS is a proposal for a new tool for data management and has nothing to do with the RDB.

Plenty of use CASE tools are currently available, but they support system development according to existing data models. The differences from CDS are mainly that
we apply a novel model, IMAH, for building CDS, and that the customer side can confirm the output by changing formulas using the defined maps after creating formulas as the input.

6. Conclusions and future work

We have developed a data processing system called the Cellular Data System (CDS) based on IMAH. In this paper, we designed algebra called unit calculation algebra on the presentation level, and added it to the functions of CDS. Using the algebra, you can calculate not only numerical values, but also unit information in the same time like you think in your head. If you take advantage of CDS in business application development, the method of development changes to visually designing spaces that express business objects. It means that the quality of a system depends on the quality of designing spaces. In application development, use of CDS will release human thinking from frequent trouble between the customer side and the supplier side. Actually, the supply management system of Maeda Corporation, for whom one of authors works, is being developed in part by using CDS.

A main future task concerns the data processing time when you use the functions of CDS. We have been surveying the relation between time and amount for data processing, and have been researching the logic for speeding up the processing time. We have gotten some good results and we will explain them in the next paper.

This research is still in its infancy, but it is progressing every day, and we are sure that CDS has possibilities to bring great social impact in the coming "Era of Contents".

References