Journal of Information Communication Engineering (JICE), 1 (1): 16-25, 2015
ISSN 2186-9162
Applied Science and Computer Science Publications

Homotopic parts configuration management using the cellular data system

Toshio Kodama, Tosiyasu L. Kunii and Yoichi Seki

Abstract: Lacking proper theory and design, big data has continued to grow in a chaotic way and are now beyond human recognition and control. To address this, we have been researching an application of the concept of homotopy preservation in homotopy type theory to remove the bottleneck in big data processing, originating from combinatorial explosion of information, which makes use of the seven layers, from homotopy to presentation, of the incrementally modular abstraction hierarchy (IMAH). We developed the Cellular Data System (CDS) as an implementation of the IMAH in our previous work. In this paper, we introduce a position information formula, which identifies the relative position of a specified factor in a formula in CDS. This function has an important role for preserving homotopy in actual changes of objects. We show the effectiveness of the formula by taking up a case study of component configuration information management in manufacturing. In the case study, it is demonstrated that design component configuration information and manufacturing component configuration information are consistently managed by preserving homotopy, even when component configuration frequently changes.

Keywords: Position information formula, Component configuration information management, Incrementally Modular Abstraction Hierarchy, Cellular Data System

I. INTRODUCTION

In Cyberworlds are information worlds formed in a cloud either intentionally or spontaneously, with or without design. As information worlds, they are either virtual or real, and can be both. In terms of information modeling, the theoretical ground for the cyberworlds is far above the level of integrating spatial database models and temporal database models. In the cyberworlds of the information age, where people are mutually connected and the activities of one person affect the behaviors of others both in cyberspace and the real world, complex systems are emerging research subjects. They are more complicated and fluid than any other previous worlds in human history, including those in the industrial age or the agrarian age, since interrelated information with exploding combinatorial complexity has to be handled and is constantly evolving. The number of organizations that conduct business in a cloud is increasing and the market is growing remarkably. On the other hand, in general business application system, as the scale of systems becomes larger and the specifications of systems changes more frequently, development and maintenance becomes more difficult, leading to higher costs and delays. In some cases, a huge system as the mainstay system in a large company, where the number of program steps is the hundreds of millions, needs several years to develop. Increases in development and maintenance cost squeeze management. For instance, it was broadcasted in Japan recently that a global vehicle manufacturer planned to spend one billion dollars to eliminate inefficiency in a huge parts list management system, which had arisen from data divergence in different component configuration information. Such situations arise because of combinatorial explosions. The era of cloud computing requires a more powerful mathematical background to model the cyberworlds and to prevent combinatorial explosions. In the cloud, every business object and business logic should be expressed in a unified form to eliminate discontinuity between systems or functions and to meet changes in user requirements. The needed mathematical mechanisms are considered to be:

1. disjoint union of topological spaces,
2. change in topological spaces while preserving invariants,
3. quotient spaces by an equivalence relation and their attachment,
4. topological space having dimensions

We consider the Incrementally Modular Abstraction Hierarchy (IMAH) that one of the authors (T. L. Kunii)

Corresponding Author: Toshio Kodama. Maeda Corporation, 2-10-2 Fujimi Chiyoda-ku, Tokyo, Japan, Tel: +81-3-3265-5551, E-mail: kodama.ts@jcity.maeda.co.jp

proposes able to satisfy the above requirements, as it models the architecture and the dynamic changes of cyberworlds from a general level (the homotopy level) to a specific one (the view level), preserving invariants while preventing combinatorial explosion [1]. It also benefits the reuse of information, guaranteeing modularity of information based on the mechanism of disjoint union. If the cyberworlds are modeled along this hierarchy, basic properties of the cyberworlds can be retained through modeling and architectural design to avoid redundant development activities and implementations. Unlike IMAH, other leading data models do not support the disjoint union or the attaching function by equivalence relation. In this research, one of the authors (Y. Seki) proposed a finite automaton called Formula Expression for the fundamental design of spaces in IMAH [9]. Another of authors (T. Kodama) has extended the expression of the automaton for wider use, which is called Extended Formula Expression, designing spaces and maps, and actually implementing a data processing system called the Cellular Data System. (Section II.) In this paper, a position information formula, which identifies the relative position of a specified factor in a formula, and its processing map, which obtains a sub-formula using the position information formula, are designed (Section III.). We demonstrate the effectiveness of the position information formula by developing general business applications for core processing of component configuration information used in manufacturing, thereby improving maintainability of the system. (Section IV.)

II. THE INCREMENTALLY MODULAR ABSTRACTION HIERARCHEY (IMAH) AND EXTENDED FORMULA EXPRESSION

A. 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 type theoretical level [11]
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. Hereafter, for simplicity, we use the term “cyberworld” and “cyberspace” interchangeably. Fiber bundles at the homotopy level define a cyberspace for a cyberworld. The product of a base space and a bundle of fibers called a fiber bundle specify a cyberspace. The product space constitutes cyberspaces. The properties defined at the homotopy level are invariants of continuous changes of functions. Homotopy is a term of Greek origin that signifies continuous deformation in a general sense. The properties that do not change by continuous modifications in time and space, such as deformation of an object and development of concepts, are expressed at this level. A cyberworld is composed of typed objects in cyberspaces. At the type theoretical level, the elements of a cyberspace are defined, and a collection of elements constitutes a type with logical operations. When we define a function in a cyberspace, we need domains that guarantee continuity such that neighbors are mapped to a near place. Therefore, a topology is introduced into the cyberspace through the concept of neighborhood. Another and equivalent way to define topology is by a power set as the strongest topology, and a null set as the weakest topology. We mainly use discrete topology in modeling information systems, and cyberworlds and cyberspaces as information worlds and information spaces. Cyberworlds are dynamic. Sometimes some cyberspaces are attached to each other and/or separated each other. These dynamic changes are described at the adjunction level. When two disjoint cyberspaces are attached, they constitute an exclusive union of two cyberspaces where attached areas of two cyberspaces are equivalent. It may happen that an attached space obtained in one way is equivalent to a space attached in another way. These attached spaces can be regarded as a set of equivalent spaces called a quotient space, which is another invariant. At the cellular structured space level, an inductive dimension is introduced into each cyberspace. At the presentation level, each space is represented in a form that may be imagined before designing a cyberworld. For example, if it is a shape, then, then geometry is used. At the view level, also called the projection level, the cyberworlds are projected onto view screens. This level has been well studied and developed as computer graphics.

B. Extended Formula Expression

1) Outline

Formula Expression is a finite automaton as a communication tool that has been developed in order to

guarantee universality in communication between subjects by expressing states of things in a formula. This is very effective in solving the frequent problems arising from misunderstandings between providers and suppliers in business application system development. Formula Expression was invented over a number of years by pursuing the greatest simplicity possible. As a result, spaces could be designed on each level of IMAH using Formula Expression, as shown in a previous paper, while this was not possible using other tools [9]. By adding an expression of square brackets [], general hierarchical modeling of things becomes possible, and by removing the definition of brackets with no elements, (), {} being equivalent to the meanings of unit element ε and zero element φ respectively, it has been extended, thus the name Extended Formula Expression. It is thought that Extended Formula Expression, by following the levels of IMAH, can reflect any space that humans create and their operations. In other words, it reflects human thought.

2) Definition

Extended Formula Expression in the alphabet is the result of finite times application of the following (1)-(8).

(1) a ($\in \Sigma^*$) is Extended Formula Expression
(2) unit element ε is Extended Formula Expression
(3) zero element φ is Extended Formula Expression
(4) when r and s are Extended Formula Expression, addition of r+s is also Extended Formula Expression
(5) when r and s are Extended Formula Expression, multiplication of r×s is also Extended Formula Expression
(6) when r is Extended Formula Expression, (r) is also Extended Formula Expression
(7) when r is Extended Formula Expression, {r} is also Extended Formula Expression
(8) when r is Extended Formula Expression, [r] is also Extended Formula Expression

Strength of combination is the order of (4) and (5). If there is no confusion, ×, (), {} can be abbreviated.

3) The generating grammar

The grammar which generates Extended Formula Expression is the following. We assume that ΣL is a set of ideograms and its element is w ($\in \Sigma$L).

G = ({E, T, F, id}, {ΣL, ε, φ, +, ×, (,), {, }, [,]}, P, E)
P = {E→T|E+T, T→F|T×F, F→(E)|{E}|[E]|id, id→w}

Here, E is called a formula, T is called a term, F is called a factor, id is called an identifier; + is called a separator, which creates a disjoint union (=addition operation) and is expressed as Σ specifically, and × is called a connecter which creates a direct product (=multiplication operation) and is also expressed as Π. The parentheses () mean a set where an order of elements is not preserved and braces {} an ordered set where the order of elements is preserved. The square brackets [] make a formula unstructured and the factor of square brackets is dealt with the same as an identifier. The factors of the parenthesis, the braces and the square brackets are called bracket factors. In short, you can say "a formula consists of an addition of terms, a term consists of a multiplication of factors, and if () or {} or [] is added to a formula, it becomes recursively the factor".

4) The Meaning of Extended Formula Expression

The language L(r) ($\in \Sigma^*$) that Extended Formula Expression r expresses is defined as:

(1) L(a) = {a} (a $\in \Sigma^*$)
(2) L(ε) = {ε}
(3) L(φ) = ∅
(4) L(r+s) = L(r)+L(s)
(5) L(r×s) = L(r)×L(s)

5) The algebraic structure of Extended Formula Expression

Extended Formula Expression r, s, t, u follow the following algebraic structure.

(1) r+(s+t) = (r+s)+t, r×(s×t) = (r×s)×t
(2) r+s = s+r
(3) r×ε = ε×r = r
(4) r×φ = φ×r = φ, r+φ = r
(5) r×(s+t) = r×s+r×t, (r+s)×t = r×t+s×t

C. The Condition Formula Processing Map as an extension function

In Formula Expression, several basic maps were defined in the previous paper [9]. An extension function for specifying conditions defining a condition formula utilizing basic maps is supported in Extended Formula Expression. This is one of the main functions, and the map is called a condition formula processing map. A formula created from these is called a condition formula. "!" is a special character which means negation. Recursivity by () in Extended Formula Expression is supported, so that the recursive search condition of a user is expressed by a condition formula. The condition formula processing map f is a map that obtains a disjoint union of terms that satisfies a condition formula from a formula. When condition formula processing is considered, the concept of a remainder of spaces is inevitable. A remainder acquisition map g is a map that has a term that doesn't include a specified factor. If you assume x to be a formula and *p, !p, p+q, p×q, !(p+q), !(p×q)* to be condition formulas, the images of *(x, p+q), (x, p×q), (x, !(p+q)), (x, !(p×q))* by *f, g* are the following:

f(x, p) = g(x, !p)

f(x, !p) = g(x, p)
f(x, p+q) = f(x, p)+f(g(x, p), q)
f(x, p×q) = f(f(x, p), q)
f(x, !(p+q)) = g(g(x, p), q)
f(x, !(p×q)) = g(f(f(x, p), q)

Cases of the condition formula including the brace {}, where the order of terms is designated, are abbreviated in this section for simplicity. In Extended Formula Expression, if you give a factor of square brackets, that is [p], in the condition formulas through the map f/g, you can get a term that does/doesn't include the factor p with perfect matching.

III. THR INTRODUCTION OF A POSITION INFOMATION FORMULA AND ITS PROCESSING MAP

I. The design of a position information formula

A position information formula is designed for specifying relative position(s) of a designated factor in a formula as an extension expression. The formula consists of a disjoint union of position information term(s) and each term consists of the multiplication of position information factor(s). The position information term is defined as follows. It is assumed that order information is:

an order of the term, which has the factor which includes a designated factor, × an order of the factor in the term.

If the factor which includes a designated factor is a bracket factor, the position information factor is:

the opening bracket + order information + the closing bracket.

If the factor which includes a designated factor equal to the designated, the position information factor is:

the opening square + order information + the closing square.

These position information factors are created recursively. Here a position information term is:

Π *a position information factor*

Then a position information formula is:

Σ *a position information term*

A simple example of the position information formula is shown below. We assume the following formula and designated factor.

formula: *a×(b+c×{d+e×f×g})+(i+j)×g*
factor: *g*

The position information formula of the factor is as follows:

(1×2)×{2×2}×[2×3]+[2×2]

Figure 1 shows the process of creating the position information formula in the example.

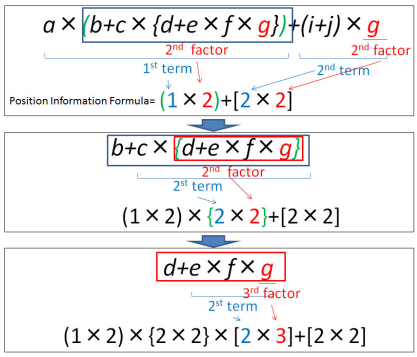

Figure 1. An example of the process of creating a position information formula

II. Sub-formula acquisition map from a position information formula f

A sub-formula acquisition map from a position information formula f is a map that obtains multiplied data as a formula with designated factors from a position information formula. This map is designed based on the condition formula processing map (II.C). If you assume the entire sets of formulas to be A, B and the entire set of terms to be C, f: B×C → A, then a designated factor p, from a position information formula pi; an arbitrary formula *x*, which doesn't include p; and arbitrary terms *r, s, t, u, v, w* follow these rules.

f: x, pi → φ
f: r×p×s+x, pi → r×p×s
f: r×(s+t×p×u+v)×w+x, pi → r×(t×p×u)×w
f: r×{s+t×p×u+v}×w+x, pi → r×{t×p×u}×w
f: r×[s+t×p×u+v]×w+x, pi → r×[s+t×p×u+v]×w

An example is shown below.

f:a×(b+c×{d+e×f×g})+(i+j)×g,(1×2)×{2×2}×[2×3]+[2×2] → a×(c×{e×f×g})+(i+j)×g

III. Implementation

The following is the coding for creating a position information formula. The focus is the recursive process (line 8) that is done if an obtained factor is a bracket type. A detailed explanation is abbreviated due to space limitations.

Name: *getPositionInformation (formula, designated factor)*

```
1    term = null, factor = null
2    while (formula is not null){
3      term = getTerm(formula)
4      while (term is not null){
5        factor = getFactor(term)
```

```
6        if(factor is a bracket factor){
7          formula = getPositionInformation(the contents,
    the factor)
8          factor = addBracket(formula)
9        } else if (factor = designated factor) {
10         factor = PositionInformationFactor
11       }
12       PI term = PI term×factor
13    }
14    PI formula = PI formula + PI term }
15  return PI formula
```

IV. CASE STUDY: DEVELOPMENT OF COMPONENT CONFIGURATION INFORMATION MANAGEMENT

A. Outline

We take up an example of the development of a core processing of component configuration information management. Component configuration information is required to manufacture a product and is critical information for manufacturing companies. There are varieties of component configuration information generally in any manufacturing industry. For example:

- Design component configuration information
  is information that is used in the design process. Classified by function, information about parts such as configurations, quantities, numbers, names, colors, weights, and so on for a finished product is registered.
- Manufacture component configuration information
  is information that is used in the manufacturing process and that corresponds to customers' requirements based on information in the above design information. This information consists of information related to production, such as places or processes for manufacturing in order to accurately assess total numbers of parts or other things.

In this section, the problems of design and manufacture component configuration information are considered briefly, due to space limitations.

Maintenance work for previously ordered products can be a source of earnings for build-to-order manufacturing companies. But the number of companies that can utilize parts list data adequately for maintenance is actually small, although those companies exploit parts list data at the design and production stages. For our purposes, assume that a product which a maker received an order for some years ago is broken. The maker searches for production-related parts list data to determine what parts are broken, but in many cases, past parts list data doesn't remain in database, so it takes a long time to check using paper documents. Moreover, even if parts of the broken product are replaced with substitutes during repair, parts list data oftentimes cannot be updated. This is because there is difficulty in consistently managing relations between design standard data and individual production data. That is, when the component configuration of a product changes (e.g. parts replacement, division of a part into subparts, set of middle components, etc.) in design component configuration information at fixed intervals for generation update or design improvement, or when parts are replaced with substitutes in manufacture component configuration information, the latest component configuration information cannot be used for maintenance, since change information in design/manufacture component configuration information data is not recorded. (Figure. 2) To solve the above difficulties, we have applied CDS to the development of core processing of component configuration information data management..

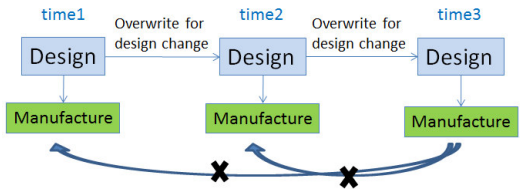

Figure 2. The reason that manufacture component configuration information cannot used for maintenance

Firstly, we make a general design of a formula for one part of a product and the replacement of the part with a substitute using CDS (IV.B). Secondly, formulas for the parts in each phase are created (IV.C). Thirdly, the formulas are processed by the maps of CDS to meet user requirements (IV.D). Here, actual data and functions are simplified to focus on verifying development of core processing without losing generality.

B. The design of a formula for design component configuration information that distinguish standard and substitute part, and processing maps

Firstly, we design a formula for parts configuration in design component configuration information as follows:

$$Design \times time_{i} \times Part_{n} \times (\Sigma(\Sigma\{Part_{n+1} \times price_{n+1} \times number_{n+1}\})) \times number_{n} + \Sigma(\$replace(position\ informat$$

ion formula+replacing Part $_m$)+$replace([1×2]+time $_j$))

where

Part $_n$:identifier of a part on level *n*, *Σ Part $_{n+1}$* are subordinates, and a first term in the braces {} means a standard part and the others are substitutes.

time: a time stamp

$replace (position information formula+identifier):

A functional identifier [10] for replacement of a specified identifier by position information formula

We call the second term and the subsequent terms *change terms*.

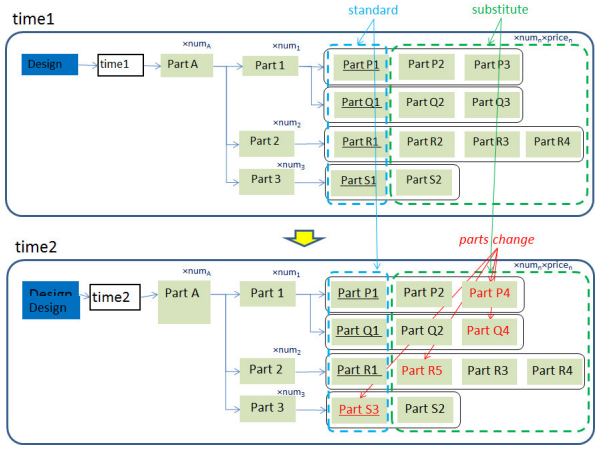

Figure 3. Examples of a formula for parts configuration and changes in design component configuration information

Secondly, we design the processing map *g* that processes the functional identifier *$replace,* which would replace a specified factor with a new factor using a position information formula in a formula, to get design configuration information at *time $_{i+1}$* below.

Design×time$_{i+1}$×Part $_m$×(Σ{Σ{Part $_{m+1}$×price $_{m+1}$×number $_{m+1}$}))×number $_m$

Simple examples of the designs are shown in Figure 3.

The formula for *time1* in Figure 3 is:

Design×time1×Part A×(Part 1×number $_1$×({Part P1+Part P2+Part P3}×{price $_{p1}$×number $_{p1}$+price $_{p2}$×number $_{p2}$+price $_{p3}$×number $_{p3}$}+{Part Q1+Part Q2+Part Q3}×{price $_{q1}$×number $_{q1}$+price $_{q2}$×number $_{q2}$+price $_{q3}$×number $_{q3}$})+Part 2×number $_2$×{Part R1+Part R2+Part R3+Part R4}×{price $_{r1}$×number $_{r1}$+price $_{r2}$×number $_{r2}$+price $_{r3}$×number $_{r3}$+price $_{r4}$×number $_{r4}$}+Part 3×number $_3$×{Part S1+Part S2}×{price $_{s1}$×number $_{s1}$+price $_{s2}$×number $_{s2}$}))

The formula for *time2* in Figure 5 is:

(The formula for time1)+($replace((1×4)×(1×2)×{1×1}×[3×1]+Part P4)+$replace((1×4)×(1×2)×{2×1}×[3×1]+Part Q4)+$replace((1×4)×{2×2}×[2×1]+Part R5)+ $replace((1×4)×{3×2}×[1×1]+Part S2)+$replace([1×2]+time2))

The figure and above formulas show parts configuration that includes information about standard parts and their substitutes, and changes from *time1* to *time2*. *Part P3, Part Q3, Part R2 and Part S1* at *time1* have been replaced with *Part P5, Part Q5, Part R5 and Part S5* at *time2* respectively. The image of the formula for *time2* in Figure 3 by the map *g* is:

Design×time2×Part A×(Part 1×number $_1$×({Part P1+Part P2+Part P3}×{price $_{p1}$×number $_{p1}$+price $_{p2}$×number $_{p2}$+price $_{p3}$×number $_{p3}$}+{Part Q1+Part Q2+Part Q4}×{price $_{q1}$×number $_{q1}$+price $_{q2}$×number $_{q2}$+price $_{q4}$×number $_{q4}$})+Part 2×number $_2$×{Part R1+Part R5+Part R3+Part R4}×{price $_{r1}$×number $_{r1}$+price $_{r5}$×number $_{r5}$+price $_{r3}$×number $_{r3}$+price $_{r4}$×number $_{r4}$}+Part 3×number $_3$×{Part S5+Part S2}×{price $_{s5}$×number $_{s5}$+price $_{s2}$×number $_{s2}$}))

The distinctive features of the above design are: 1. the hierarchical structure of parts configuration, which can be expressed in the formulas as it is, and 2. the functional identifier *$replace(...)*, through which information about changes is expressed as addition of terms. As a result, the formula for parts configuration can roll back to any state quite easily.

Thirdly, we also design a formula for parts configuration with production information in manufacture component configuration information as follows:

Manufacture×time $_i$×Part $_n$×(Σ Assembly $_{n+1}$×Place $_{n+1}$×Part $_{n+1}$)

where Part $_n$, Assembly $_n$, Place $_n$ are identifiers of a part, an assembly, and a place on level *n* respectively.

Next, we design the calculation map *f* to calculate prices and numbers [10]. The image of the formula for *time 1* (using standard parts) by the map *f* is:

number $_1$×(price $_{p1}$×number $_{p1}$+price $_{q1}$×number $_{q1}$)+number $_2$×price $_{r1}$×number $_{r1}$+number $_3$×price $_{s1}$×number $_{s1}$

C. Data input

We assume design component configuration information of car manufacturing and the changes in parts configuration from *time1* to *time2*, as shown in Figure 4. It is assumed that there are *Engine 1, Body 1, Steering wheel 1, Brake 1 and Lamp 1* as standard components and others are their substitutes for *CAR A* at *time1*, that *Brake 1* is substituted for *Brake 4*, and that *Engine 1* is separated into *E-part 1* and *E-parts 2* at *time2*. A price and a number for each part are

assumed as you see in Figure 5. The formula for the above situation is as follows:

Formula E:

Design×time1×(CAR A×({Engine 1+Engine 2+Engine 3}×{1×50+1×45+1×40}+{Body 1+Body 2}×{2×40+2×35}+Steering wheel1×1×30+{Brake 1+Brake 2+Brake 3}×{2×20+2×30+2×30}+{Lamp1+Lamp2}×{10×4+5×4}))+($replace((1×3)×(1×2)×{4×1}×[1×1]+Brake 4)+$replace((1×3)×(1×2)×{4×2}×[1×1]+2×40)+$replace((1×3)×(1×2)×[1×4]+(E-parts 1+E-parts 2))+$replace([1×2]+time 2))

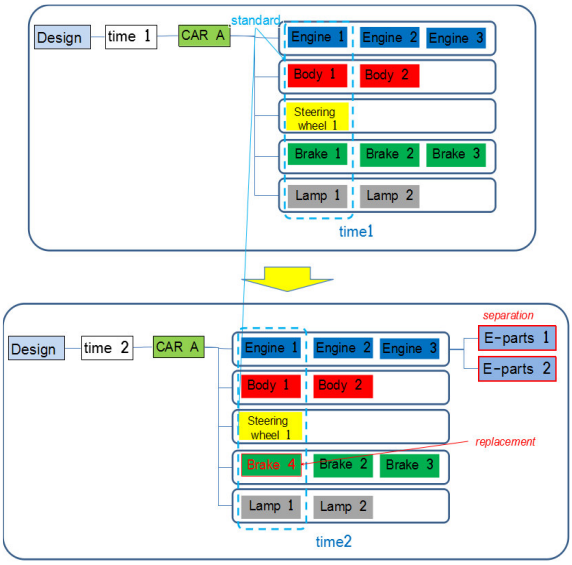

Figure 4. An example of design component configuration information at time1and time2

Part Id	Price	Num	Part Id	Price	Num
Engine 1	50	1	Brake 3	30	2
Engine 2	45	1	Brake 4	40	2
Engine 3	40	1	Lamp 1	10	4
Body 1	40	2	Lamp 2	5	4
Body 2	35	2	...	...	...
Steering wheel1	30	1			
Brake 1	20	2			
Brake 2	30	2			

Figure 5. An example of prices and numbers of parts

Next, it is assumed that manufacture component configuration information at *time1* is created based on the design component configuration information by adding production process information as shown in Figure 6. The formula (*Formula M1*) for it is created according to the above design.

Formula M1:

Manufacture×time1×(CAR A×(a1×{place1+place2+place3+place4}×{Engine 1+Body 1+Steering wheel 1+component X×a2×place5×{Brake 1+Lamp 2}×{1×50+2×40+1×30+{2×20+4×5}}}))

In the same way, it is assumed that manufacture component configuration information data at time2 is created based on the design component configuration information, as shown in Figure 7. The formulas (*Formula M2*) for them are created according to the above design:

Formula M2:

Manufacture×time2×(CAR A×(a1×{place1+place2+place3+place4}×{Engine 1×a3×place6×{E-parts 1+E-parts 2}+Body 1×a4×place7×{B-parts 1+B-parts 2}+Steering wheel 1+component X×a2×place5×{Brake 1+Lamp 1}}))

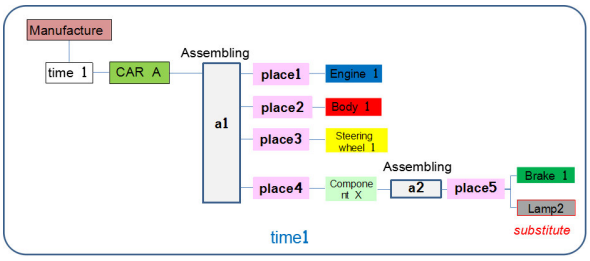

Figure 6. An example of manufacture component configuration information at time1

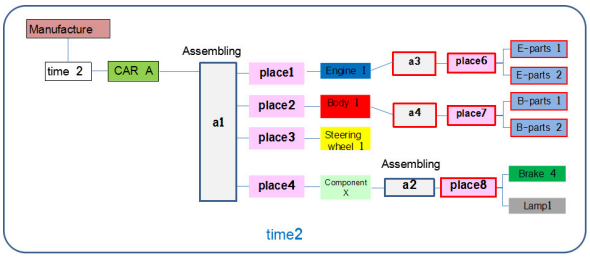

Figure 7. An example of manufacture component configuration information at time2

D. Data output

We take up some examples of user requirements according to the problems of data inconsistency mentioned in the *Outline*. (IV.A)

First, when you want to know how the part *Lamp 1* at *time1* in design component configuration information changed in manufacture component configuration information, you first get the formula for design component configuration information data at *time1* from *Formula E* by removing the change terms. Secondly, you get sub-formulas of the factor of *Lamp 1*

or its substitute *Lamp 2* from a disjoint union of the result and *Formula M1* through the conditional formula processing map and attach the sub-formulas by equivalent factors through the attachment map [9]. Calculation of numbers is also done by the calculation map in the same time. This yields the following formula (Figure. 8):

{Design+Manufacturing}×time1×CAR A×{ε+a1×place1×component X×a2×place5}×{Lamp 1+Lamp 2}×{40+20}

You can know from the result that the process creating an intermediate *component X* which includes *Lamp 2* was added and *Lamp 2* was substituted for *Lamp 1* at *time1* in manufacture component configuration information and that this substitution brings cost savings of *20* (*=40-20*).

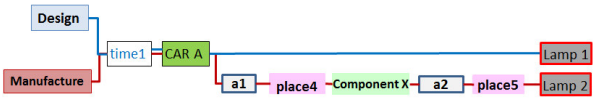

Figure 8. The correspondence of *Lamp* in design/manufacture component configuration information at time1

Next, when you want to know what the parts *B-parts 1* at *time2* in manufacture component configuration information was in design component configuration information at *time2*, you first get the formula for design component configuration information at *time2* from *Formula E* using map *g* (IV.B). If you create a disjoint union of the formula and *Formula M2* and attach them by equivalent factors through the attachment map [9], you get the following formula (Figure. 9).

{Design+Manufacturing}×time2×CAR A×{ε+a1×place2}×Body 1×{ε+a4×place7}×B-parts 1}

You can know from the result information about the process of creating an intermediate *B-parts 1* at *time2* in manufacture component configuration information.

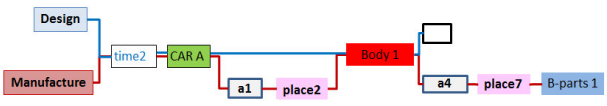

Figure 9. The correspondence of *B-parts 1* in design/management component configuration information at time2

Next, when you want to know how production information of the part *Brake* in manufacture component configuration information changed from *time1 to time2*, you first get the position information formula of the factor *Brake 1* from *Formula M1* as follows:

(1×3)×(1×2)×{1×3}×{4×4}×[1×1]

Then you get the formulas from *Formula M1* and *Formula M2* through the sub-formula acquisition map by the position information formula and attach them by equivalent factors and calculate them as follows (Figure. 10):

Manufacture×{time1+time2}×(CAR A×(a1×place4×component X×a2×{place5+place8}×{Brake 1+Brake 4}×{40+80}))

You can know from the result that the part *Brake 1* at *time1* changed to the part *Brake 4* at time2, which brings the cost increase of *40* (*=80-40*), and the place for assembling changed from *place5* at *time1* to *place8* at *time2* in manufacture component configuration information.

Figure 10. The correspondence of *Brake* in manufacture component configuration information at time1/time2

E. Considerations

In the above examples, in design component configuration information, component configuration is designed as a formula and changes to it are designed as the addition of a functional identifier *$replace* to it. By doing so, you can roll back the formula for design component configuration information quite easily to any previous state of the past and you can also get the formula for the latest state of configuration by using the processing map of the functional identifier. In manufacture component configuration information, each formula for manufacture component configuration information at each time is created based on each formula for the design component configuration information, and a standard part and its substitute part are expressed as the brace {} where orders of elements are kept. Thereby, the correspondence of parts between design component configuration information and manufacture component configuration information at any time can be known using the condition formula processing map, the attachment map and the calculation map, and the correspondence between manufacture component configuration information at different times can be known using the position information formula, its processing map and the other maps. We consider the recognition of changes of component configuration as homotopy preservation. As you see in the above examples, the typical problems of data inconsistency in

component configuration information management, which is mentioned in the Outline (IV.A), can be solved so that component configuration information can be exploited for maintenance. Furthermore, even when different part codes for the same part are managed between design/manufacture component configuration information data, the attachment function can be applied by adding information of equivalence relation in the same way. This way of data modeling, where information about component configuration and its changes are reflected as they are, can reduce the number of development of application programs and improve maintainability of the system

V. 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 structured space extends the topological space, as seen in Section 2. Relational OWL as a method of data and schema representation is useful when representing the schema and data of a database [2][4], 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 does not 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 [7] aims at minimizing document revalidation in an XML schema evolution, based in part on the graph theory. The X-Entity model is an extension of the Entity Relationship (ER) model and converts XML schema to a schema of the ER model. In the approach of [5], the conceptual and logical levels are represented using a standard UML class and the XML represents the physical level. XUML [7] 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 the cellular model. That is because the tree structure, on which the XML model is based, and the graph structure [3], 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 cannot in general be expressed by XML.

Although CDS and the existing deductive database apparently look alike, the two are completely different. The deductive database [8] raises the expression capability of the relational database (RDB) by defining some rules. On the other hand, CDS is a new tool for data management, and has nothing to do with the RDB.

VI. CONCLUSION

We have developed a data processing system called the Cellular Data System (CDS) modeling the Incrementally Modular Abstraction Hierarchy (IMAH). In this paper, we introduced a position information formula and its processing map which obtains a sub-formula using the position information formula in CDS. We demonstrated in the case study that, if you use this function with the other functions of CDS in component configuration information management, data consistency can be guaranteed, since you can effectively determine correspondence between parts or components in different component configuration information based on information about their relative positions in the parts configuration. Consequently, business application development becomes simpler, business processes more visible, and business applications more flexible to changes in variable business conditions. Furthermore, use of CDS can make developers more creative, while preventing combinatorial explosions and reducing the need for intensive testing. This function has already been put to practical use in many companies in Japan, and further strides are being made every day.

REFERENCES

[1] T. L. Kunii and H. S. Kunii, “A Cellular Model for Information Systems on the Web - Integrating Local and Global Information”, *In Proc. of DANTE'99*, IEEE Computer Society Press, pp.19-24, 1999.

[2] Perez de Laborda, C. ; Conrad, S., “Bringing Relational Data into the SemanticWeb using SPARQL and Relational.OWL”, Proceedings. 22nd International Conference on Data Engineering Workshops, 2006. IEEE, pp.55.

[3] David W. Embley, “Semantic priming in a cortical network model”, *Journal of Cognitive Neuroscience*, MIT Press, vol.21 Issue12, pp.2300-2319, 2009.

[4] Ernestas Vysniauskas, Lina Nemuraite, “TRANSFORMING ONTOLOGY REPRESENTATION FROM OWL TO RELATIONAL DATA”, Information Technology and Control, Kaunas University of Technology, vol.35, pp.333-343, 2006.

[5] Yuan An, John Mylopoulos, Alex Borgida, “Building semantic mappings from databases to ontologies”, *In Proc. of AAAI'06*, AAAI Press, pp.1557-1560, 2006.

[6] Bernadette Farias Lósio, Ana Carolina Salgado, Luciano do Rêgo GalvÐo, “Conceptual Modeling of XML Schemas”, *In Proc. of WIDM'03*, ACM Press, pp.102-105, 2003.

[7] Stephen J. Mellor, Marc J. Balcer, "Executable UML: A Foundation for Model Driven Architecture", Addison Wesley, 2002.

[8] Faiz Arni , Kayliang Ong , Shalom Tsur , Haixun Wang , Carlo Zaniolo, "The Deductive Database System *LDL++*", *Theory and Practice of Logic Programming*, Cambridge University Press, pp.61-94, 2003.

[9] Toshio Kodama, Tosiyasu L. Kunii, Yoichi Seki, "A New Method for Developing Business Applications: The Cellular Data System", *In Proc of CW'06*, pp. 65-74, IEEE Computer Society Press, 2006.

[10] Toshio Kodama, TosiyasuL. Kunii, Yoichi Seki, "The Development of Core Logic for an Estimate System using the Cellular Data System", *In Proc of CW'11*, pp. 211-216, IEEE Computer Society Press, 2006.

[11] Don Monroe, "A New Type of Mathematics?", Communications of the ACM, Vol. 57 No. 2, Pages 13-15, 2014.2