Reprint: Tosho Kodama, Tosiyasu L. Kunii, Yoichi Seki, "An Example of a Tracking Function Using the Cellular Data System", Proceedings of the 9th WSEAS International Conference on ARTIFICIAL INTELLIGENCE, KNOWLEDGE ENGINEERING and DATA BASES (AIKED'10), Murray Edwards College, University of Cambridge, UK, February 20-22, 2010 , pp. 372-377.

An Example of a Tracking Function Using the Cellular Data System

TOSHIO KODAMA[1], TOSIYASU L. KUNII[2], YOICHI SEKI[3]
[1] CDS Business Dept., Advanced Computer Systems, Inc. and Maeda Corporation, Tokyo, JAPAN
kodama@lab.acs-jp.com, https://www.cellulardatasystem.com/e/index.html
[2] Morpho, Inc., The University of Tokyo, Tokyo, JAPAN
kunii@ieee.org, kunii@acm.org, http://www.kunii.net/
[3]Software Consultant, Tokyo, JAPAN
gamataki61@mail.hinocatv.ne.jp

Abstract: In the era of cloud computing, where data and data dependencies constantly change, a mechanism within system development that can correspond to those changes in user requirements is needed. The Incrementally Modular Abstraction Hierarchy (IMAH) offers the most appropriate mathematical background to model dynamically changing information worlds by descending from the abstract level to the specific, while preserving invariants. In this paper, we have applied the Cellular Data System (CDS), based on IMAH, to the development of core logic for a budget tracking function, and verified that using CDS makes the data modeling simpler.

Key-Words: incrementally modular abstraction hierarchy, formula expression, topological space, tracking function

1 Introduction

Cyberworlds are more complicated and fluid than any other previous worlds in human history, and are constantly evolving and expanding. One of the features of cyberworlds is that data and its dependencies are constantly changing within them. For example, millions of users communicate with each other on the Web using mobile devices, which are considered one of the main elements of cyberworlds. At the same time, user requirements for cyberworlds also change and become more complicated as cyberworlds change. If a user analyzes data in business applications correctly under a dynamically changing situation using the existing technology, the schema designs of databases and application programs have to be modified whenever schemas or user requirements for output change. That leads to combinatorial explosion. To solve the problem, we need a more powerful mathematical foundation than what current computer science enjoys. As a possible candidate, we have introduced the Incrementally Modular Abstraction Hierarchy (IMAH), built by one of the authors (T. L. Kunii). IMAH seems to be the most suitable for reflecting cyberworlds, because it can model the architecture and the changes in cyberworlds and real worlds from a general level to a specific one, preserving invariants while preventing combinatorial explosion [1]. In our research, one of the authors (Y. Seki) has proposed an algebraic system called Formula Expression as one of finite automaton, and another (T. Kodama) has designed how to express the spaces and the maps on each level of IMAH and actually implemented IMAH as a data processing system using Formula Expression [6]. We call the system the Cellular Data System (CDS). CDS has already been applied to the development of several business application systems as a flexible system development tool. In this paper, we have applied an attaching function to file permission management. The attaching map was defined on the adjunction level of IMAH [1] and the attaching function is based on the map; terms as topological spaces are attached by common factors found through the attaching function. If the function is used with the condition formula search that is the main data search function of CDS (2.3), it becomes a very effective means of analyzing data in cyberworlds without losing consistency in the entire system, since a user can get the data desired without changing application programs. In addition, we put emphasis on practical use by taking up an example of the development of a file permission information management system. First, we explain IMAH and Formula Expression briefly (Section 2). Next, we demonstrate the effectiveness of CDS by developing core logic of a budget tracking function system (Section 3). Related works are mentioned (Section 4), and, finally, we conclude (Section 5).

Reprint: Tosho Kodama, Tosiyasu L. Kunii, Yoichi Seki, "An Example of a Tracking Function Using the Cellular Data System", Proceedings of the 9th WSEAS International Conference on ARTIFICIAL INTELLIGENCE, KNOWLEDGE ENGINEERING and DATA BASES (AIKED'10), Murray Edwards College, University of Cambridge, UK, February 20-22, 2010 , pp. 372-377.

2 The Cellular Data System

2.1 Incrementally Modular Abstraction Hierarchy

The following list constitutes 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. The properties defined at the homotopy level are invariants of continuous changes of functions. 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 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 each other, 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, which is another invariant. 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.

2.2 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 ε is Formula Expression
(3) zero element φ 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×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, ×, (), {} can be abbreviated. + means disjoint union and is expressed as specifically and × is also expressed as Π. 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, five maps (the expansion map, the bind map, the division map, the attachment map, the homotopy preservation map) are defined [9].

2.3 A Condition Formula Search

A function for specifying conditions defining a condition formula by Formula Expression is supported in CDS. This is one of the main functions. A formula created from these is called a condition formula. "!" is a special factor which means negation. Recursivity by () in Formula Expression is supported so that the recursive search condition of a user is expressed by a condition formula. A condition formula processing map *f* is a map that gets 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. Fig 1 shows each image by the condition formula processing map *f*.

2.4 Implementation

This application was developed using Java Servlet and Tomcat 5.0 as a Web server. The specifications of the server were:

OS: Red Hat Enterprise Linux 5
CPU: Intel Core2 Duo (1.8GHz)
RAM: 4GB
Web server: Apache 2.2.2
AP server: Tomcat 5.5.4
JAVA: JDK1.5.015
RDB: mysql5.1

Reprint: Tosho Kodama, Tosiyasu L. Kunii, Yoichi Seki, "An Example of a Tracking Function Using the Cellular Data System", Proceedings of the 9th WSEAS International Conference on ARTIFICIAL INTELLIGENCE, KNOWLEDGE ENGINEERING and DATA BASES (AIKED'10), Murray Edwards College, University of Cambridge, UK, February 20-22, 2010 , pp. 372-377.

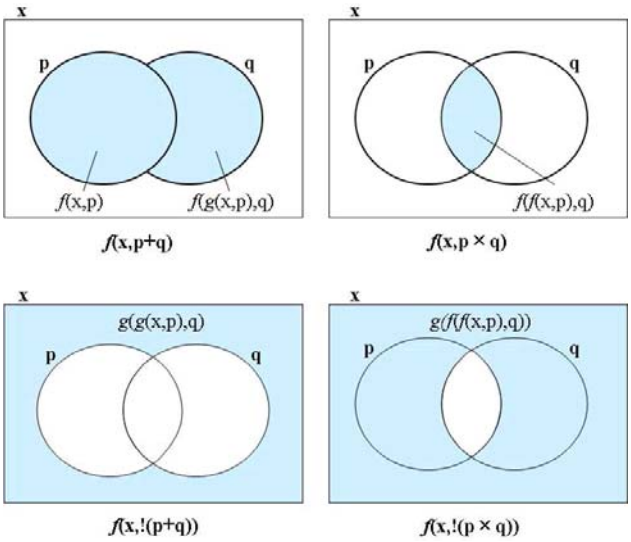

Fig 1 Images by the condition formula processing map f

HD: 240GB
The specifications of the client machine were:
OS: Windows XP
CPU: Intel Core2 Duo (3.00GHz)
RAM: 4GB

A quotient acquisition map is the main function of a condition formula processing map. In this algorithm, the absolute position of the specified factor by the function of the language and the term including the factor are acquired first. Next, the nearest brackets of the term are acquired and, because the term becomes a factor, a recursive operation is done. Details are abbreviated due to the restriction on the number of pages.

3 A Tracking Function as an Application

3.1 Outline

We have developed a business application of the core logic for budget flow tracking using CDS. In this system, the budget is managed under the assumption that the budget flows from upstream to downstream, while user records business data in slips that have different attributes, relating them each other using a pointer attribute (Fig 2). In this section, we simplify all data without losing generality. Firstly, we design a formula for a cellular space of slips and processing to obtain pointer information of the slips. Secondly, we use the maps when we output the data according to user requirements.

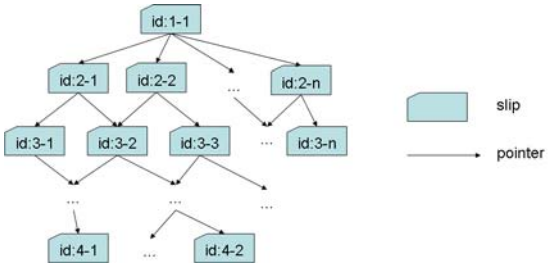

Fig 2 flow of budgets slips

3.2 The design of space and processing

We design a formula for the cellular space as follows:

1. A term for a slip as a cellular space

$slip_i\{\mathrm{PT}+\Sigma attribute_j\}(\Sigma id_k\{upper\ slip\ id_k+\Sigma value_j\})$

$slip_i$: a factor which expresses a slip name
PT: a factor which expresses a pointer attribute
$upper\ slip\ id_k$: a factor which expresses id of an upper slip.

A formula for slips is expressed as a disjoint union of terms for a slip.

2. Processing to obtain pointer information from the formula

A formula for pointer information as a topological space is obtained through the condition formula processing map (2.3) from the formula for slips.

$f(formula\ for\ slips, \text{'PT'})$
$= \Sigma slip_i \times \mathrm{PT}(\Sigma(id_k \times upper\ slip\ id_k))$

3.3 Data input/output

Here, budget slip data is to be managed. Assume that the budget flows through four steps from upstream to downstream: plan, order, delivery, and payment. Slips are made in each step, where the id value of a slip higher up is given the pointer attribute "*PT*", as seen in Fig 3 and Fig 4. If a user wants to input slip data, he/she creates formula 3.3-1 according to the space design.

Reprint: Tosho Kodama, Tosiyasu L. Kunii, Yoichi Seki, "An Example of a Tracking Function Using the Cellular Data System", Proceedings of the 9th WSEAS International Conference on ARTIFICIAL INTELLIGENCE, KNOWLEDGE ENGINEERING and DATA BASES (AIKED'10), Murray Edwards College, University of Cambridge, UK, February 20-22, 2010 , pp. 372-377.

step	Attributes
plan	PT,PlanName,Budget
order	PT,OderName,UnitPrice,Amount,Company,Date
delivery	PT,ProductName,Compay,TotalPrice,Amount,Date
payment	PT,PaymentName,Company,Price,Date

Fig 3 Attributes of the slips in each step

Slip: plan1

PT	PlanName	Budget
-	projectA	100

Slip: order1

PT	OrderName	UnitPrice	Amonunt	Price	Company	Date
plan1	o1	15	2	30	c1	date1

Slip: order2

PT	OrderName	UnitPrice	Amonunt	Price	Company	Date
plan1	o2	30	2	60	c2	date2

Slip: delivery1

PT	OrderName	Company	Price	Date
order1	d1	c3	10	date3

Slip: delivery2

PT	OrderName	Company	Price	Date
order1	d2	c4	15	date4

Slip: delivery3

PT	OrderName	Company	Price	Date
order2	d3	c5	40	date5

Slip: payment1

PT	PaymentName	Company	Price	Date
delivery1	p1-1	c5	5	date6
delivery2	p1-2	c4	5	date7

Slip: payment2

PT	PaymentName	Company	Price	Date
delivery2	p2-1	c5	5	date7
delivery3	p2-2	c4	40	date8

Fig 4 The slips during each step

formula 3.3-1:
Plan{PT+PlanName+Budget}(plan1{ε+ProjectA+100})+Order{PT+OrderName+UnitPrice+Amount+Price+Company+Date}(order1{plan1+o1+15+2+30+c1+date1}+order2{plan1+o2+30+2+60+c2+date2})+Delivery{PT+ProductName+Company+Price+Date}(delivery1{order1+d1+c3+10+date3}+delivery2{order1+d2+c4+15+date4}+delivery3{order2+d3+c5+40+date5})+Payment{PT+PaymentName+Company+UnitPrice+Date}(payment1({delivery1+p1-1+ company4+5+date6}+{delivery2+p1-2+company2+5+date7})+payment2({delivery2+p2-1+c5+5+date7}+{delivery3+p2-2+c4+40+date8}))

If a user wants to answer the question "What is the budget flow in the slips?", firstly he/she gets the image of formula 3.3-1 by the factor "*PT*" through the map *f* according to the processing design. From the results, you will obtain the budget flow shown in Fig 5.

f (*formula3.3-1*,'*PT*')
=Plan×*PT*(plan1{ProjectA})+Order×*PT*(order1{plan1}+order2{plan1})+Delivery×*PT*(delivery1{order1}+delivery2{order1}+delivery3{order2})+Payment×*PT*(payment1({delivery1}+{delivery2})+payment2({delivery2+delivery3}))
={Plan+Order+Delivery+Payment}*PT*{plan1+(order1×plan1+order2×plan1)+(delivery1×order1+delivery2×order1+delivery3×order2)+(payment1(delivery1+delivery2)+payment2(delivery2+delivery3))}
(formula 3.3-2)

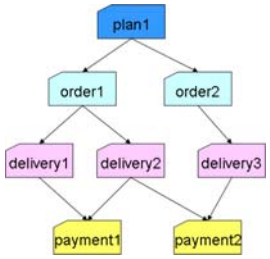

Fig 5 Budget flow within a company

Next, if a user wants to answer the question "How much has not yet been delivered in the slip for *order1*?", firstly he/she gets the image of formula 3.3-2 and the factor "*order1*" through the map *f*.

f (*formula 3.3-2*, *"order1")*
=Order×PT(*order1*×plan1+(delivery1+delivery2)*order1*)

From the result, you can see that the budget of *order1* flows from *plan1* and flows to *delivery1* and *delivery2*. And secondly, he/she gets the image of formula 3.3-1, by *"order1×Price"*, *"delivery1×Price"* and *"delivery2×Price"* through the map *f* respectively.

f (*formula 3.3-1*, *"order1×Price")*
=Order×Price×order1×**30**

f (*formula 3.3-1*, *"delivery1×Price")*
=Order×Price×*delivery1*×**10**

f (*formula 3.3-1*, *"delivery2×Price")*
=Order×Price×*delivery2*×**15**

From these, you can see that prices of *order1, delivery1, delievry2* are *30, 10 and 15* respectively,

Reprint: Tosho Kodama, Tosiyasu L. Kunii, Yoichi Seki, "An Example of a Tracking Function Using the Cellular Data System", Proceedings of the 9th WSEAS International Conference on ARTIFICIAL INTELLIGENCE, KNOWLEDGE ENGINEERING and DATA BASES (AIKED'10), Murray Edwards College, University of Cambridge, UK, February 20-22, 2010 , pp. 372-377.

and that the price of the *order1* which has not yet been delivered is 5 (=30-10-15) by simple calculations.

3.4 Considerations

In existing business application development of a budget flow tracking function under the assumption that the format of slips often changes and budget flows unexpectedly, it is generally difficult and costly to develop and maintain the system. On the other hand, if CDS is employed instead, the design of data structure as formulas is more adaptable to changes (such as in slip format or budget flow) and the design of the tracking function of business objects becomes quite simple using the map, if you design the cellular space and design the processing of necessary information from the created formula, and it therefore becomes simple and economical to develop and maintain the function. In other words, the business objects and their relations, and business logic are described directly and simply by CDS and the data that a user wants are outputted in parts from the inputted data through the maps of CDS.

4 Related works

The distinctive features of our research are the application of the concept of topological processing, which deals with a subset as an element, and that the cellular space extends the topological space, as seen in Section 2. The conceptual model in [2] is based on an ER model, where tree structure is applied. The approach in [3] aims at grouping data of a graph structure where each node has attributes. The ER model, graph structure and tree structure are expressed as special cases of topological space, and a node with attributes is expressed as one case of the cellular space. These models are included in the function of CDS. Many works dealing with XML schema have been done. The approach in [4] aims at introducing simple formalism into XML schema definition for its complexity. An equivalence relation of elements is supported in CDS, so that complexity and redundancy in schema definition are avoided if CDS is employed, and a homotopy preservation function is introduced into CDS in the model for preserving information. As a result, the problems described in [4] do not need to be considered in CDS. Some works of inductive data processing have been done recently. CDS can also be considered as an inductive systems. The goal of research on the inductive database system of [7] is to develop a methodology for integrating a wide range of knowledge generation operators with a relational database and a knowledge base. The main achievement in [8] is a new inductive query language extending SQL, with the goal of supporting the whole knowledge discovery process, from pre-processing via data mining to post-processing. If you use the methods in [7], [8], the attributes according to users' interests have to be designed in advance. Therefore it is difficult to cope with changes in users' interests. If you use CDS, a formula for a topological space without an attribute as a dimension in database design can be created so that the attributes of objects don't need to be designed in advance.

5 Conclusions

In this paper, we have applied CDS to the development of the core logic of the budget flow tracking function in a company. If you use CDS during development, application programs can become simple because of its flexibility at the time of modeling. The point we should emphasize is that the quality of the system using CDS is closely related to how the formulas for space are designed according to IMAH [1]. The design of formulas is fully various, because Formula Expression is very simple in describing business objects and their relations. Therefore, the creativity of the system developer designing the formulas becomes more important when he/she uses CDS.

References:

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

[2] Anand S. Kamble, "A conceptual model for multidimensional data", In Proceedings of APCC'08, Tokyo, Japan, Australian Computer Society, Inc., pp.29-38, 2008.

[3] Alexandr Savinov, "Grouping and Aggregation in the Concept-Oriented Data Model", In Proceedings of SAC'06, Dijon, France, ACM press, pp.482-486, 2006.

[4] Wim Martens, Frank Neven, Thomas Schwentick, Geert Jan Bex, "Expressiveness and complexity of XML Schema", ACM Transactions on Database System, pp.770-813, 2006.

[5] Denilson Barbosa, Juliana Freire, Alberto O. Mendelzon, "Designing Information-Preserving Mapping Schemes for XML", In Proceedings of

VLDB'04, Trondheim, Norway, VLDB Endowment, pp.109-120, 2004.

[6] Toshio Kodama, Tosiyasu L. Kunii, Yoichi Seki, "A New Method for Developing Business Applications: The Cellular Data System", In Proceedings of CW'06, Lausanne, Switzerland, IEEE Computer Society Press, pp.64-74, 2006.

[7] Kenneth A. Kaufman1, Ryszard S. Michalsk, Jarroslaw Pietrzykowski, and Janusz Wojtusiak, "An Integrated Multi-task Inductive Database VINLEN: Initial Implementation", Knowledge Discovery in Inductive Database, LNCS 4747, pp.116-133, Springer-Verlag Berlin Heidelberg, 2007.

[8] S. Kramer,V. Aufschild, A. Hapfelmeier, A. Jarasch, K. Kessler, S. Reckow, J. Wicker and L. Richter "Inductive Databases in the Relational Model: The Data as the Bridge", Knowledge Discovery in Inductive Database, LNCS 3933, pp.124-138, Springer-Verlag Berlin Heidelberg, 2006.