LEGO ROBOT DESIGN USING INCREMENTALLY MODULAR ABSTRACTION HIERARCHY

Kenji Ohmori
Computer and Information Sciences
Hosei University
3-7-2 Kajino-cho
Koganei-shi, Tokyo, Japan
email: ohmori@hosei.ac.jp

ABSTRACT
Incrementally Modular Abstraction Hierarchy (IMAH) is a new design method that consists of abstraction levels starting from the most general level and ending at the most specific one. IMAH is a common method that can be applied to a wide variety of application areas. In this paper, IMAH is applied to an embedded system that is implemented as a concurrent system. In a concurrent system, each component of the system is regarded as an agent. The agents in the system cooperate with each other to achieve required services. This paper describes the design of the concurrent system that is theoretically supported by IMAH while avoiding logical faults. The design of a LEGO robot is carried out by descending the abstraction hierarchy, where the specification is transformed to components, state transition diagram, the description of communicating sequential processes (CSP) and program codes. When descending the abstraction hierarchy, a homotopy lifting property (HLP), which is one of the most important properties in homotopy theory, is used when applying a bottom-up approach. In contrast, a homotopy extension property (HEP) is used for a top-down approach. The design method has succeeded in providing a theoretical approach, which enables to implement a secure system.

KEY WORDS

1 Introduction

Generalization and specification are common ideas in designing a system of any field. When designing a system, you may draw an image of a system on some abstraction level and then concrete it or generalize it. When developing a software system in Java, we may use UML as an abstract description. In hardware design, you may use a schematic diagram. As these methods share common properties, we have considered to unify them and proposed the incrementally modular abstraction hierarchy (IMAH) [1], in which the abstraction hierarchy starts from the homotopy level. In mathematics, homotopy theory [2], [3], [4], which unifies algebra and geometry, is the most abstract concept. The IMAH is constituted by with mathematical abstraction hierarchy. The IMAH has been applied to many fields such as an accounting system [5], Japanese house architecture [6], logical thinking [7], cyberworlds [8] and computer graphics [9].

Many kinds of machines are now implemented as embedded systems that are controlled by multiple computers and provide sophisticated services. The paper describes how to apply the IMAH to designing an embedded system. As the embedded system is characterized as a system to receive many signals through sensors, decide next behavior and send signals to external devices, the embedded system has to carry out many processes concurrently according to events that occur inside and outside of the system. Communicating sequential processes (CSP) [10], which can directly handle concurrency and events, is more suitable for the implementation of the embedded system.

A LEGO robot is designed to explain how the IMAH is applied to a real system though it is not so complicated. A LEGO robot is equipped with a control device, which is called NXT [11], installing an ARM processor. In our design, the NXT is replaced by an XMOS processor that is an event driven and multi-thread processor. An XMOS processor [12] contains up to 32 cores, each of which executes one process. A traditional processor like ARM realizes multi processes by time-slicing in a preemptive multi-tasking system, in which inputs and outputs cause interruption. Actually, only one process runs at a time. On the other hand, an XMOS processor truly runs multi-processes concurrently. Inputs and outputs of an XMOS processor are treated as events. By combining multiple XMOS processors, it is possible to execute a huge number of processes concurrently. The LOGO robot that is studied in this paper is rich in scalability. A more sophisticated robot that has a huge number of processors as well as many inputs and outputs is realized by adding a large number of XMOS processors.

When designing a system, composition and decomposition are important design methods along with generalization and specification. The homotopy lifting property (HLP) and homotopy extension property (HEP) give a mathematical foundation. The HLP enables a bottom-up...
method for composition and the HEP a top-down one for decomposition. The bottom-up method is realized by attaching two subsystems. When attaching two subsystems, the combined system is contaminated by logical faults if the interfaces of two subsystems do not match. In the conventional system, as the relation of the interfaces is not defined mathematically, it is hard to avoid logical faults. However, in our design method, along with the HLP and HEP, an adjuncting map that combines two subsystems mathematically guaranties the correctness of the combined system.

2 Mathematical foundations for developing a LEGO robot

Our designing approach is characterized by generalization and specification achieved by the IMAH and bottom-up and top-down methods constituted by a pullback and a pushout. Therefore, the approach is funded strongly by mathematical concepts.

2.1 The incrementally modular abstraction hierarchy

The IMAH is constructed by the abstraction hierarchy starting from the most abstract level to the most specific one.

- **The homotopy level**: The most fundamental shape of the structure of the developing system is defined including the number of connected spaces and the fundamental group of each space. Hardware components and software agents in the developing system are treated as separate spaces.

- **The set theoretical level**: Each space is configured as a set. The functions of a hardware component or the services of a software agent become elements of its set.

- **The topological space level**: A topology is induced into each set. The set becomes a topological space, which gives a strong mathematical foundation when designing the system.

- **The adjuncting space level**: The static and dynamical behavior of the developing system is clarified so that relations among hardware components and software agents (separated spaces) are defined.

- **The cellular space level**: Realistic images of a hardware component or a software agent is clarified using CW complexes, in which a hardware component or a software agent is configured as an n-dimensional entities. A CW complex represents a state transition diagram in a sophisticated way.

- **The presentation level**: This level is the starting point in the traditional architecture and modeling. CSP is defined for the software agents and hardware components.

- **The view level**: Finally, program codes and logical circuits are obtained.

2.2 The homotopy lifting property and homotopy extension property

The mathematical foundations for the HLP and HEP are summarized as follows.

- **Adjuncting space**: Let us start with a topological space \(X\) and attach another topological space \(Y\) to it. Then, \(Y_f = Y \sqcup_f X = Y \sqcup X/\sim\) is an adjoining space obtained by attaching \(Y\) to \(X\) by an adjoining map \(f\) (or by identifying each point \(y \in Y_0\) \(| Y_0 \subset Y\) with its image \(f(y) \in X\) by a continuous map \(f\)). \(\sqcup\) denotes a disjoint union. The adjoining map \(f\) is a continuous map such that \(f : Y_0 \to X\), where \(Y_0 \subset Y\). Thus, the adjoining space \(Y_f = Y \sqcup X/\sim\) is a case of quotient spaces \(Y \sqcup X/\sim = Y \sqcup_f X = Y \sqcup X/(x \sim f(y) \forall y \in Y_0)\).

A matchmaking party gives a good example to explain how separate groups are merged together using an adjoining map. There are a group of girls, which is represented by the set \(G = \{Betty, Michel, Alice, Chelsea, Jackie\}\), and another group of boys described by the set \(B = \{Tom, Mike, Peter, Jack, Alex\}\). A member of girls has never met a member of the boy group. Suppose that these groups have a matching party. The situation before the meeting is represented by the disjoint union of two groups. The situation after the meeting is described using an adjoining map, which shows new partners that have been made at the party.

The disjoint union of these two sets are represented by \(G \sqcup B = \{(Betty, f), (Michel, f), (Alice, f), (Chelsea, f), (Jackie, f)\}, (Tom, m), (Mike, m), (Peter, m), (Jack, m), (Alex, m)\}\.

At the meeting, Betty and Mike like each other and become a partner. Also, Chelsea and Peter become another partner as shown in Figure 1. If a partner is considered to be identical by an adjoining map \(f\), then, \(G \sqcup_f B = \{(Betty, f), (Mike, m), (Michel, f), (Alice, f), (Chelsea, f), (Peter, m), (Jackie, f), (Tom, m), (Jack, m), (Alex, m)\}\ where \(f : G_0 \to B\) and \(G_0 = \{(Betty, f), (Chelsea, f)\}\).

- **Pullback**: A pullback is the limit of a diagram consisting of the two morphisms \(f : X \to Z\) and \(g : Y \to Z\) with a common codomain (target), where a morphism is a function in set theory and a continuous map in topology. Conventionally, the limit is given by the set \(\{(x, y) \in (X, \times Y)|f(x) = g(y)\}\), where an element is the product of two spaces. Instead, we use an adjoining space, which is the disjoint union of two spaces, since a pullback combines two disjoint spaces into one space by attaching these
common spaces. The concept of a pullback becomes a mathematical foundation of a bottom-up approach.

- **Pullback:** A pullback is the limit of a diagram consisting of the two morphisms \( f : Z \to X \) and \( g : Z \to Y \) with a common domain (source). A pullback breaks the attached (common) space \( Z \) into the two disjoint spaces \( X \) and \( Y \). The concept of a pullback gives a mathematical foundation of a bottom-up approach, which can be used also as a divide-and-conquer method.

The matching party gives another example of a pullback. If \( Z = G \sqcup_f B \), then, \( X = B \) and \( Y = G \) give a pullback. \( Z \) is the disjoint union (colimit) of \( X \) and \( Y \) as shown in Figure 2.

![Figure 2](image2.png)

Figure 2. A pullback and pushout are the mathematical foundation of bottom-up and top-down approaches.

- **Fiber Bundle:** A fiber bundle is another example of a pullback. A fiber bundle is constructed by the total space \( E \), the projection \( p \), which maps \( E \) to the base space \( B \), and the fiber \( F \) where \( E \) locally looks like a product space of \( B \times F \).

An example is shown in Figure 3. A cylinder of \( E \) is pushed out to a ring of \( B \) and fibers (vertical lines) of \( F \). By applying it inversely, a pullback is obtained. In Figure 3, a Mobius strip of \( E \) is pulled back by a ring of \( B \) and fibers (rotating vertical lines) of \( F \).

- **Homotopy:** Continuous maps \( p, q \) are homotopic if there exists a continuous map \( H : [0, 1] \times I \to Y \) such that \( H(x, 0) = p(x) \) and \( H(x, 1) = q(x) \), where \( I \) is the unit interval \([0, 1]\). \( H \) is called homotopy of \( p \) and \( q \), denoted by \( p \simeq q \).

![Figure 3](image3.png)

Figure 3. A fiber bundle gives a pullback and pushout.

- **Path:** A continuous map \( \lambda : I \to X \) yields a path. \( \lambda(0) = x \) and \( \lambda(1) = y \) are called the initial and terminal points. The path is denoted by \( w = (W, \lambda) \) where \( W = \lambda(I) \). The path space on \( X \), denoted \( X^I \), is the space \( \{ \lambda : I \to X \text{ continuous} \} \) endowed with the compact-open topology.

- **Homotopy Lifting Property:** A homotopy lifting property (HLP) is a generalization of a fiber bundle by changing the property of \( p \) from a projection to a continuous map.

The HLP is defined as follows. Given any commutative diagram of continuous maps as shown in Figure 4, the map \( p : E \to B \) has the homotopy lifting property if there is a continuous map \( \hat{H} : Y \times I \to E \) such that \( \hat{H} \times I_0 = h \) and \( p \circ \hat{H} = H \). The homotopy \( \hat{H} \) thus lifts \( H \) through \( p \) and extends \( h \) over \( I_0 \) where \( I_0(a) = (a, 0) \). A continuous map \( p : E \to B \) is a fibration if \( p \) has the homotopy lifting property.

As \( \hat{H} : Y \times I \to E \) and \( p^{-1} : B \to E \) give a pushout, the HLP can be used as a tool for a bottom-up approach.

- **Homotopy Extension Property:** The homotopy extension property is dual to the homotopy lifting property. Given any commutative diagram of continuous maps as shown in Figure 4, there is a continuous map \( \hat{K} : X \to Y^I \) such that \( p_0 \times \hat{K} = k \) and \( \hat{K} \times i = K \). The homotopy \( \hat{K} \) thus extends \( K \) over \( i \) and lifts \( k \) through \( p_0 \) where \( p_0(\lambda) = \lambda(0) \). In this case, \( i \) has the homotopy extension property. An inclusion of a closed subspace \( i : A \hookrightarrow X \) is a cofibration if \( i \) has the homotopy extension property. \( Y^I \) is the path space on \( Y \).

As \( \hat{K} : X \to Y^I \) and \( \iota^{-1} : X \to A \) give a pushout,
3 Designing a LEGO robot

Table 1 shows design summary on each abstraction level. As the LEGO robot design is composed by hardware and software design, there are corresponding summaries for each of them. As hardware design shares some properties with software design, these properties are described in the common column.

Suppose we develop a line trace car with a XMOS process, two photo sensors and two motors. The line trace car runs on a black line. Usually, the black line lies between the two photo sensors, which are located at the either outside of the black line. If a photo sensor detects the black line, the car is crossing the black line at the same side of the sensor. Assuming that the line trace car is controlled intuitively, the line trace car stops the wheel locating at the same side of the sensor that detects the black line.

The IMAH is flexible to allow starting any level of the abstraction hierarchy. Then, we can descend or ascend the abstraction hierarchy to design more abstractly or concretely. When descending or ascending the abstraction hierarchy, we can use bottom-up and top-down methods, which are endowed by mathematical concepts of pullback and pushout. In this paper, the line trace car is developed using the agile development method, where the system is developed repeatedly. The most valuable part is firstly developed. Then, the second valuable part is developed after the completion of the first part. This process is repeated until the system development is completed. Therefore, we start to development the line trace car system from the homotopy level and concrete some part by descending the abstraction hierarchy. When finishing the development the first part, this development process is repeated.

3.1 The homotopy level

The homotopy level for the hardware design is achieved by providing 5 connected spaces as shown in Figure 5, each of which stands for the processor or a peripheral device. The fundamental group of each space is not clear at this moment. Some space may be connected to another space. However, the connection of spaces is not decided until some concrete level design is completed.

Figure 5. Homotopy level design gives the fundamental shape of structure for hardware components and software agents.

The homotopy level design for software provides 4 independent agents at this moment, each of which takes care of the allocated peripheral device. These agents are shown in Figure 5.

3.2 The set theoretical level

On this level, a software agent or hardware component is more concreted by providing services or functions that it provides. At certain stage of the development, some services (functions) are clear and others are not. Others will be clarified later in advancement of designing. A set is accommodated to each agent (function). Clarified services (functions) become an element of the set. Others also become one element of the set. When others are clarified, it may be broken down into several elements.

Suppose that we are designing a Sensor agent, the agent has services to the Photo sensor and the Motor agent and other services, which are not clear at this moment. As the main function, the agent receives a signal from the Photo sensor and sends it to the Motor agent. The set for a Sensor agent is constituted by \{Photo, Motor, Others\}.

The space of a sensor agent is the disjoint union of Photo space, Motor space and Others space, which is described as SensorAgent = \{Photo \sqcup Motor \sqcup Others\}.

For hardware components, it is also designed in the same way.
Table 1. The abstraction hierarchy for LEGO robot design

<table>
<thead>
<tr>
<th>IMAH</th>
<th>Common</th>
<th>Hardware</th>
<th>Software</th>
</tr>
</thead>
<tbody>
<tr>
<td>Homotopy</td>
<td># of connected spaces, fundamental groups</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Set theory</td>
<td>A set of functions for each component</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Topological space</td>
<td>Topological graph</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Adjoining space</td>
<td>State transition diagrams</td>
<td>Hardware interface</td>
<td>Channel (Message passing)</td>
</tr>
<tr>
<td>Cellular space</td>
<td>Modified state transition diagrams</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Presentation</td>
<td>Domain specific high level description</td>
<td>HDL</td>
<td>CSP</td>
</tr>
<tr>
<td>Projection</td>
<td>Domain specific low level description</td>
<td>Logical circuits</td>
<td>Program codes by XC</td>
</tr>
</tbody>
</table>

3.3 The topological space level

On this level, the relation among services or functions is clarified. In particular, a sequence of services (functions), which shows which service (function) is carried out before another service (function), is defined on this level. The sequence of services can be represented as a graph in such a way a service (or function) becomes a vertex and a transition from one service to another becomes an edge. The graph is represented by a set of vertexes and the edges so that \( \{a | a \in V \cup E \} \), where \( V = \{v_i\} \) is a set of the vertexes and \( E = \{e_k = <v_j,v_k>\} \) is a set of the edges. The topology is induced by the power set of the graph. Suppose that the length of each edge is the unit interval, the distance is also introduced to the topological space. The graph extended to topology is called the topological graph theory [13].

For a Sensor agent, we have the following set \( \{v_1 = Photo, v_2 = Motor, v_3 = Others, e_1 = <v_1, v_2>, e_2 = <v_2, v_1>\} \)

By providing edges, the space of a Sensor agent is changed to \( SensorAgent = Photo \sqcup_f Motor \sqcup Others \), where the adjoining map \( f \) attaches \( Photo \) to \( Motor \) by identifying the transitions.

![Figure 6. A sequence of services is represented using a topological graph.](image)

3.4 The adjoining space level

On this level, the state transition diagram is defined for each service or function. An example is shown in Fig 7. At the Photo service of a Sensor agent, it receives an OFF or ON signal, each of which shows that the Photo Sensor detects the black line or does not. Therefore, the Photo service is divided into two parts; one receives for an OFF signal, the other for an ON signal. Each of them has two state and transit from one state to the other state by receiving a signal. In one case, it receives an OFF signal and sends STOP to the Motor agent. In the other case, it receives an ON signal and sends RUN to the Motor agent. These two cases are combined by a bottom-up approach or a pullback by attaching the identical states.

![Figure 7. The state transition diagram is obtained using a pushout and pullback.](image)

A service of an agent is related with a service (function) of another agent (or a component). The Photo service of a sensor agent receives signals from the Photo Sensor. If the transition diagram of the photo sensor is obtained as shown at the upper part of Figure 8, it is connected to the Photo service using a pullback where the identical edges are attached by an adjoining map as shown in Figure 8.
3.5 The cellular space level

On this level, the conditions of a state transition are converted to vertexes so that the state transition graph is transformed to a CW complex. An example is shown in Figure 9. The state transition diagram is changed to a CW complex along the interval $I$. In this example, $I$ is a sequence of edges, each of which represents the condition of a state transition and is converted to a CW complex. When forming a CW complex at $E$, $B$ has to be preserved, where $B$ is the set theoretical space. In this example, $B$ is $\{\text{PhotoSensor, SensorAgent}\}$. If $I$ is the sequence of formalizations from the bottom edge to the top edge, we have the two layers of CW complexes as shown in Figure 9. The first one clarifies the bottom edge. The second one does the bottom and top edges.

The CW complex that is obtained by the above process is described in Figure 10. The CW complex consists of eleven 0-dimensional, fifteen 1-dimensional and three 2-dimensional open balls. Using the filtration space, the CW complex is constructed step by step. Firstly, the 0-dimensional open balls are provided. Then, the 1-dimensional open balls are provided and each 1-dimensional open ball is attached to its boundary balls. For example, $e^1_0$ is attached to $e^0_0$ and $e^0_0$ by an adjuncting map $f$: $\partial e^1_0 \to e^0_0 \cup e^0_0$, or $e^1_0 \sqcup f(e^0_0 \cup e^0_0)$. Finally, the 2-dimensional open balls are provided and each of them is attached its boundary balls. In this example, the CW complex of the Sensor agent is connected to the Photo sensor through the two 1-dimensional open balls, which sends signals from the Photo sensor to the Sensor agent.

3.6 The presentation level

On this level, the cellular space is converted to the presentation space along the interval while preserving the set theoretical space. The conversion is carried out using the HLP for a bottom-up method. In the HLP, $H(X, I)$ maps $X$ to $E$ along $I$. $H(X, I)$ is described by the continuous map $f_t(X)$ where $t \in I$. $f_t$ converts the CW complex into CSP. When $t$ changes from the start point of the Interval to the end point, the area in the CW complex that is changed to CSP is expanded from null to the whole. Conversion rules are described in Figure 11.

- If two 0-dimensional open balls are connected by a single 1-dimensional open ball, a sequence of the two processes, which represent the 1-dimensional open balls, is obtained.
- If multiple 0-dimensional open balls are connected sequentially by 1-dimensional open balls, each pair of 0-dimensional open balls is converted to a sequence of two processes step by step. Finally, a sequence of these processes is obtained.
- If there are multiple CW complexes, each of which is converted as a single process. These processes are executed in parallel, which is described by a symbol of $||$.
- If two dimensional balls are connected by multiple 1-dimensional open balls, the multiple processes, each of which is an alternative process, are provided. Each process is connected by a symbol $[]$.

An example is shown in Figure 12. $X$ is the CW complex obtained by the above process. The CW complex represents the state transition diagrams of the Photo Sensor and the Photo service in the Sensor agent. $I$ is the order of conversion to CSP. In the example, the CW complex of...
Figure 10. The CW complex is obtained to give a concrete image to a designer.

Figure 11. Conversion rules are applied when obtaining CSP. The Photo sensor is firstly transformed to CSP. Then, the CW complex of the Photo service is converted. $B$ is the set of two elements: the Photo sensor and the Sensor agent. These elements are mapped to $E$ by $p^{-1}$ so that the process diagrams that are obtained along $I$ are related to the Photo sensor and Sensor agent.

3.7 The view level

Program codes, which are written by using XC, are obtained from CSP. XC is a C-like language, but has functions of CSP. XC allows providing parallel processes that communicate each other through channels that send and receive messages. A separate processor is allocated permanently for each process. As events from inputs and to outputs are directly handled in a program, a peripheral device can be easily controlled in a program.

The conversion from CSP to program codes is carried out in a same way as previously described.

- $\text{processA} | | \text{processB}$ is converted as parallel processes using a par statement.
  
  \[
  \text{par} \{
  \text{processA}();
  \text{processB}();
  \}
  \]

- $\text{processA}[\text{processB}]$ is converted as alternative processes using a switch statement.

The program codes obtained from CSP are shown in Figure 13. In the figure, the correspondences between CSP and the program codes are explained step by step. It is known that the program codes are almost automatically generated from CSP. The relation between the cellular space level (the CW complex) and the view level (the program codes) can be also shown easily by mapping each open ball of the CW complex to the corresponding program part.

3.8 Agile software development

The main advantage of agile software development is repetitive developments in which the most valuable part is firstly developed, the second one is then developed after completion of the first part and this development process is repeated until the system development is completed. In the early phase of development, agile software development allows for expectation of its success.

The development process described above is repeated using agile software development until the system development is completed. The IMAH enhances the ability of agile software development by giving mathematical foundation.
Following the above development, the Photo service of a Sensor agent may be more clarified. The others service will be changed to the log service that records inputs from the Photo sensor. Starting from the homotopy level, the logo service is added on the set theoretical level and the execution sequence along with the Photo and Motor services is defined. The lower level design descending from this level is also carried out in the same way.

So far, the HEP has not been applied. However, the Sensor agent is divided into the logo service and the services that have already designed so that the HEP is used when designing the logo service.

4 Conclusion

The LEGO robot has been successfully designed on the mathematical foundation, which is constituted by a pullback, pushout, homotopy extension property and homotopy lifting property. When descending the abstraction hierarchy, the invariants that are defined on an upper level are preserved on a lower level, which guarantees design correctness among different levels.

The IMAH, which give mathematical foundation for generalization and specification, allows providing a flexible design method, where a developer can start any level of design, generalize it by ascending the abstraction hierarchy and specifying it. The pushout and pullback give another mathematical foundation for top-down and bottom-up approaches. Logical errors that contaminates to the developing system when applying the conventional approach are avoided by the mathematical foundation.

As the transformation from one abstraction level to a neighbor abstraction level is autonomous, the design process can be automated, which helps to provide a new computer aided design system for software development backed by the mathematical foundations.

References


