Formalization of Distributed and Dynamic Resources Allocation Using Category Theory

Zhen You[a,b,*], Jinyun Xue[a,b], Ying Shi[b], Dongming Jiang[b], Qimin Hu[a]
[a] Provincial Key Laboratory for High-Performance Computing Technology,
Jiangxi Normal University, Nanchang, 330022, China
[b] State Key Laboratory of Software Engineering, Wuhan University, Wuhan, 430072, China
Email: youzhen.jxnu@gmail.com

Abstract— **Distribution and dynamics are the main characteristics of resource allocation problem. Despite the variety of approaches and models proposed for the problem, a systematic formalization and a general solution strategy are missing. This paper takes a step towards this goal by proposing a categorical formalization of resource allocation, which not only represents both distributed and dynamic features, but also formally prove the properties of symmetry, safety(non-deadlock), liveness(non-starvation) and concurrency. The application to the wireless sensor networks demonstrates the practicability of our formal model and emphasizes the strengths of category theory – its simple theory, abstract mechanism, diagrammatical representation and its expressive power in representing and reasoning many concepts of computer science.**

Index Terms— **Resource Allocation, Category Theory, Formal Model, Wireless Sensor Networks**

I. INTRODUCTION

THE widespread development of high-speed networks and the availability of powerful computers at low-cost have led to the much research about distributed and dynamic resource allocation problem. The problem arises in many real-world domains such as wireless sensor networks, emergency equipment/personnel management, disaster rescue and hospital scheduling. Resource allocation may be decided by using different models[1,2,3,4,5] and algorithms[6,7,8] applied to a specific domain to automatically and dynamically allocate resources to applicants.

Historically, category theory stemming from algebraic topology is a way to provide a basic conceptual and notational framework, which is useful to demonstrate various structural concepts of different fields in a uniform way. There is a large body of work in category theory ranging from purely categorical studies to applications of categorical principles in different fields. As an important tool, category theory impinges more and more frequently on the awareness of many computer scientists[9], especially those with an interest in formal specifications and semantics-based research. The significance of category theory in theoretical computing science was further confirmed by the work of Smyth-Plotkin[10] and Lehmann-Smyth[11]. Another nice examples are Lehmann's[12] concerns the important of using powerful theorems from category theory and Goguen-Burstall's[13] use of certain theorems about colimits constructions.

Category theory is a useful formal framework in computer science. First of all, because of the focus that is put in relationship (categorical morphisms) and not in entities (categorical object). Secondly, it provides abstraction and generality ideas – abstracting from unnecessary details to give general definitions and results, and focusing on how things behave, rather than on what their internal details are. Another, the fact of the matter is that category theory is essentially algebraic means that it can be used to reason about structure and the mappings that preserve structure naturally arises in many different areas of computing. With the above advantages, we believe that category theory can contribute with the formal model of distributed and dynamic resource allocation in the same way as it contributed in the areas of design and implementation of programming languages[14,15], type theory[16], specification language[17], automata theory[18,19], architecture[20,21], models of concurrency[22,23], and syntax and semantic model [24,25].

A. Previous work

Theretofore, several attempts have been made to find a satisfactory formal model to resource allocation problem. Modi and Jung[1] proposes a formalization of distributed resource allocation and a general solution strategy using distributed constraint satisfaction techniques. A self-stabilizing deterministic model and algorithm[2]–the main idea is the construction of a spanning tree for each processes – is developed, and the correctness and performance of the model is formally proven.An automatic resource allocation strategy[3] based on market mechanism is proposed to achieve resource balance in cloud computing. A new theoretic resource allocation model based on ESPSA(Extended Second Price Sealed Auction)

Manuscript received April 18, 2013; revised *****; accepted *****.

This work was supported in part by the National Nature Science Foundation of China (Grant No. 61272075), the major international cooperative research project (Grant No.61020106009) from National Natural Science Foundation of China, the National Natural Science Foundation of Jianxi Province (Grant No.20132BAB211022) and the Science and Technology Research Project of Jianxi Province Educational Department(Grant No.GJJ13231).

doi:10.4304/jsw.8.12.3088-3095

is developed to guarantee task's victorious probabilities on competing for limited resources. The novel spatial-based Network-on-Chip resource allocation algorithms can be used to efficiently reduce the communication congestion and improve the overall performance.

The famous solution for distributed resource allocation problem is based on an acyclic directed graph technique which was presented by K. M. Chandy and J. Misra[5]. This strategy can be efficiently guarantee safety, liveness and fairness of the classical dining philosophers problem, the dynamic drinking philosophers problem[26] and the committee coordination problem[27].

B. Main contribution

Despite the variety of approaches and models proposed for distributed or dynamic resource allocation, a systematic formalization of the problem and a general solution strategy are missing. This paper takes a step towards this goal by proposing a categorical formalization of resource allocation, which not only represents both dynamic and distributed aspects of the problem, but also formally prove the properties of symmetry, safety(non-deadlock), liveness(non-starvation) and concurrency. The formal models defined in the paper is based on Chandy-Misra's acyclic directed graph strategy [5,27] and our previous experience[28] in implementing the categorical semantics for distributed dinning philosophers problem. Finally, we describe how our categorical models can be implemented in wireless sensor networks.

Another goal is bring the benefits of the use of category theory to the field of formal model for distributed and dynamic resource allocation: not only formalizing usual concepts in resource allocation using categorical entities (such as objects, morphisms, colimits), but also giving good directions to reason/preserve the universal structure and facilitate its application to practical problems. Meanwhile, the abstraction mechanism, diagrammatical representation and powerful expressive ability of category theory make it easier to clarify the traceability and understandability of resource allocation problem in distributed and dynamic environment.

C. Paper Structure

The paper is structured as follows: Section 2 presents some basic definitions of category theory. Sections 3 formalizes the categorical models for distributed and dynamic resource allocation, and show how these categorical concepts and operations (such as Sums, Pushouts and Colimits) used to merge, detect similarities and hide details of the problem. Meanwhile, its correctness and properties are also elaborated. In section 4, we extend the formal models to handle a generalized problem in wireless sensor networks. Concluding remarks and future work are finally discussed in Section 5.

II. Category Theory

What is category theory? There are many different answers to this question, but generally speaking: Category theory is a branch of mathematics that has been developed over the last 50 years, and is concerned with the study of algebraic structure. In the section, some fundamental definitions about category theory needed in the rest of the paper are reviewed. Most of them are excerpted from Jaap van Oosten's book[29], and José Luiz Fiadeiro' book[30].

Definition 1. ***Category*.**

A category $\mathcal{C}$ is given by a collection $\mathcal{C}_0$ of objects and a collection $\mathcal{C}_1$ of morphisms which have the following structure. Each morphism in $\mathcal{C}_1$ has a ***domain*** and a ***codomain***, which are objects in $\mathcal{C}_0$; one writes $f : x \to y$ if x is the domain of the morphism f, and y its codomain; One also writes $x = dom(f)$ and $y = cod(f)$. The category $\mathcal{C}$ must satisfied the following laws:

- ***Identity law:*** For every object x there is an identity morphism $id_x : x \to x$, satisfying $id_x \circ g = g$ for every $g : y \to x$ and $f \circ id_x = f$ for every $f : x \to y$;
- ***Composition law:*** Given two morphisms $f : x \to y$ and $g : y \to z$ such that $cod(f) = dom(g)$, the composition of f and g, written $g \circ f : x \to z$, is defined and has domain $dom(f)$ and codomain $cod(g)$;
- ***Associative law:*** Composition is associative, that is: given $f : x \to y$, $g : y \to z$ and $h : z \to w$, $h \circ (g \circ f) = (h \circ g) \circ f$.

Remark 1. Categories are special kinds of graphs. As demonstrated in the Fig. 1, a category is simply a labeled graph with "composite" and "identity" edges that satisfy the obvious equational laws.

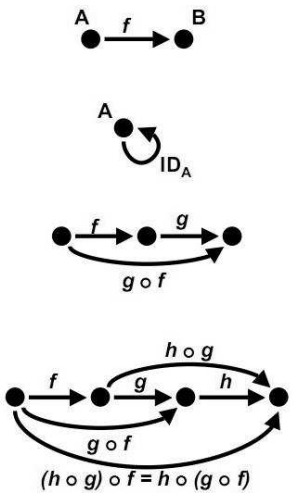


Figure 1. Graphic Description of Category

Definition 2. ***Graph homomorphism*.**

Let $\mathcal{G}$ and $\mathcal{H}$ be graphs. A homomorphism of graphs $\varphi : \mathcal{G} \to \mathcal{H}$ is a pair of maps $\varphi_0 : \mathcal{G}_0 \to \mathcal{H}_0$ and $\varphi_1 : \mathcal{G}_1 \to \mathcal{H}_1$ such that for each arrow $f : x \to y$ of $\mathcal{G}$ we have $\varphi_1(f) : \varphi_0(x) \to \varphi_0(y)$ in $\mathcal{H}$.

Definition 3. ***Diagram*.**

Let $\mathcal{C}$ be a category and $\mathcal{I}$ a graph. A diagram in $\mathcal{C}$ with shape $\mathcal{I}$ is a graph homomorphism $\delta : \mathcal{I} \to graph(\mathcal{C})$.

Definition 4. ***Commutative Diagrams*.**

Let $\mathcal{C}$ be a category, its diagram $\mathcal{D}$ is said to commute iff, for every pair x,y of nodes and every pair of paths

TABLE I.
UNIVERSAL CONSTRUCTIONS

Construction	Co-Construction
Terminal Objects	Initial Objects
Products	Sums
Pullbacks	Pushouts
Equalizers	Coequalizers
Limits	Colimits

$W = U_1, U_2, ..., U_m$, $W' = V_1, V_2, ..., V_n$ from x to y in diagram $\mathcal{D}$,

$$U_m \circ U_{m-1} \circ ... \circ U_1 = V_n \circ V_{n-1} \circ ... \circ V_1$$

holds in category $\mathcal{C}$.

Definition 5. ***Colimit.***

Let $\delta : \mathcal{I} \to graph(\mathcal{C})$ be a diagram in category $\mathcal{C}$. A colimit of δ is a commutative cocone $p : \delta \to z$ such that, for every other commutative cocone $p' : \delta' \to z'$, there is a unique morphism $f : z \to z'$ such that $f \circ p = p'$.

Remark 2. There are various universal constructions (see Table 1) in categories. Each construction has a dual, formed by reversing the morphisms. Colimit is a generalization of other universal constructions, which includes initial object, sum(coproduct), pushout and coequalizer.

Definition 6. ***Functor.***

Let $\mathcal{C}$ and $\mathcal{D}$ are two categories, a functor $\mathbb{F} : \mathcal{C} \to \mathcal{D}$ consists of operations $\mathbb{F}_0 : \mathcal{C}_0 \to \mathcal{D}_0$ and $\mathbb{F}_1 : \mathcal{C}_1 \to \mathcal{D}_1$, such that

- for each morphism $f : x \to y$ in category $\mathcal{C}$, there is a morphism $\mathbb{F}_1(f) : \mathbb{F}_0(x) \to \mathbb{F}_0(y)$ in category $\mathcal{D}$;
- for each object x in category $\mathcal{C}$, the equation $\mathbb{F}_1(id_x) = id_{\mathbb{F}_0(x)}$ holds in category $\mathcal{D}$;
- for each pair of morphisms $f : x \to y$ and $g : y \to z$ in category $\mathcal{C}$, the equation $\mathbb{F}_1(g \circ f) = \mathbb{F}_1(g) \circ \mathbb{F}_1(f)$ holds in category $\mathcal{D}$.

Remark 3. A functor is a category of categories. In fact, its objects are categories and its morphisms are certain structure-preserving maps between categories[31]. On the other hand, A functor is simply a labeled graph homomorphism that also preserves the identity and composition edges that are present in categories.

III. FORMALIZATION OF RESOURCES ALLOCATION

The Resource Allocation Problem consists of 1) **a set of resources** that may be shared by two or more tasks, and 2) **a set of tasks**, which request a subset of resources to access to the **Critical Section (CS)** of code and perform the necessary actions. For each task there is a set of neighbor tasks. Each task has a conflict with its neighbors: it cannot share the CS with any of them.

The problem of resolving conflicts between tasks in distributed systems is of practical importance. In this section, we present the formal model and diagrammatical solution of distributed and dynamic resource allocation problem using category theory, which not only efficiently resolves the deadlock and starvation, but also ensure the satisfaction of some properties.

A. Signature

The signature of a resource allocation problem is a structure $\Theta = \{\Upsilon, \Gamma, \Lambda\}$ where

- Υ is a set of resource, $\Upsilon = \{\Upsilon_1, \Upsilon_2, ..., \Upsilon_m\}$.
- Γ is a set of tasks, $\Gamma = \{T_1, T_2, ..., T_n\}$.
- Λ is a set of actions, $\Lambda = \{\Lambda_1^1, \Lambda_2^1, ..., \Lambda_p^i, \Lambda_n^j\}$. Each task contains a set of actions, where Λ_p^i denotes $i'th$ action of task Γ_p.

The communication between conflict tasks is only of two kinds: a task either requests the permission to execute the CS from the neighbors, or releases this permission. A task is in one of 3 states: 1) ***idle***, 2) ***request*** and 3) ***running***.

B. Categorical Model of Distributed Resource Allocation

The basic idea behind our solution to resource allocation is a variant of Chandy-Misra's acyclic directed graph strategy[5,27]. We define for each task T_i a set of **conflict neighbors** $T_i.F$ and a set of **communication neighbors** $T_i.C$. Both relations are symmetric. That is, for any two tasks T_i and T_j if $T_i \in T_j.F$ then $T_j \in T_i.F$. The same applies to $T_i.C$. We do not consider asymmetric communication.

An application of the conflict and communication relations is in the management of files in the distributed environment. Each file is represented a resource. Two tasks are conflict neighbors that if that one of them write into a file that they share; two tasks are communication neighbors that both of them only read common files.

The general **distributed resource allocation problem**[26] states there are n symmetrical tasks $\Gamma = \{T_1, T_2, ..., T_n\}$ (with identical protocols) contending for access to m resources $\Upsilon = \{\Upsilon_1, \Upsilon_2, ..., \Upsilon_m\}$. Each task requires a fixed subset of the resources. Both concurrent access and exclusive access are encountered. Each task continually cycles between its critical section and its noncritical section. A solution to this problem guarantees each task is eventually able to enter its critical section and access its needed resources with the resource constraints satisfied.

A conflict between two or more tasks must be resolved in favor of some (usually one) task and against the other conflict tasks: a favored task must have priority over others. A distributed implementation of an **acyclic precedence graph**, where the task with the lower precedence must yield to the process with greater precedence in finite time, is a conflict resolution rule ensures fair resolution of all conflicts. Precedences among tasks are typical **partial ordering** relations, so it easier to model with category theory.

Definition 7. ***Precedence Category:*** $\mathbb{P} = \{\Gamma, \Psi\}$**.**

The precedence category $\mathbb{P}$ is composed of a objects collection $\Gamma = \{T_1, T_2, ..., T_n\}$ in signature, each object represents a task; and a morphisms collection Ψ, where for every morphism $f_{ij} : T_i \to T_j \in \Psi$ $(T_i, T_j \in \Gamma \wedge T_i \neq T_j)$ means that task T_i has priority over task T_j, and for every object T_i in Γ,there is an *identity* morphism

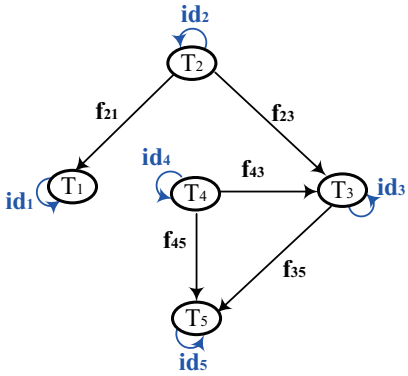


Figure 2. Diagram of precedence Category

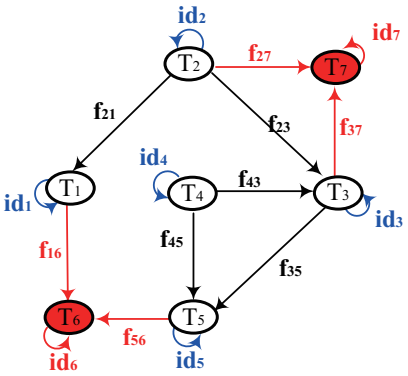


Figure 3. Adding Tasks

$id_i : T_i \to T_i$, which indicates that the priority of T_i is equivalent to itself.

Proof of Precedence Category:

- Given two morphisms $f_{ij} : T_i \to T_j$ and $f_{jk} : T_j \to T_k$ such that $dom(f_{ji}) = cod(f_{ij})$. The *composition* of f_{ij} and f_{ji}, written $f_{ji} \circ f_{ij} : T_i \to T_k$, is defined and has domain $dom(f_{ij})$ and codomain $cod(f_{ji})$. The composition means the precedence of tasks are ***transitive***;
- Composition is ***associative***, that is, given $f_{ij} : T_i \to T_j$, $f_{jk} : T_j \to T_k$, and $f_{kw} : T_k \to T_w$, then $f_{kw} \circ (f_{jk} \circ f_{ij}) = (f_{kw} \circ f_{jk}) \circ f_{ij}$;
- Each ***identity*** morphism $id_i : T_i \to T_i$ satisfies $id_i \circ f_{ji} = f_{ji}$ for every $f_{ji} : T_j \to T_i$, and $f_{ij} \circ id_i = f_{ij}$ for every $f_{ij} : T_i \to T_j$.

Category theory supports the diagrammatic representation which visualizes the relationships between concepts. It is possible to use diagrams to express and reason about precedence among tasks in a formal way.

Example 1: The diagram of precedence category $\mathbb{P}$ with 5 tasks in distributed resource allocation problem is given in Fig. 2.

Remark 4. The diagram of precedence category $\mathbb{P}$ is commutative.

The commutative property of a diagram helps to establish a set of equalities between morphisms. Hence, diagrams and commutativity provide us with the ability of doing equational reasoning in a visual form.

C. Categorical Model of Dynamic Resource Allocation

As a variant of distributed resource allocation problem, **dynamic resource allocation problem**[26] allows the number of tasks to fluctuate as needed. This allows tasks can be added or deleted from the problem, and tasks can add and delete resources.

In order to solve general dynamic resource allocation problems that conform to our formalized model– ***Precedence Category***, four rules are defined as follows.

Rule 1. ***Adding Task:*** New task have lower precedence to their conflicting neighbors.

Example 2: Two new tasks T_6 (shares resource with T_1, T_5) and T_7 (shares resource with T_2, T_3) are added

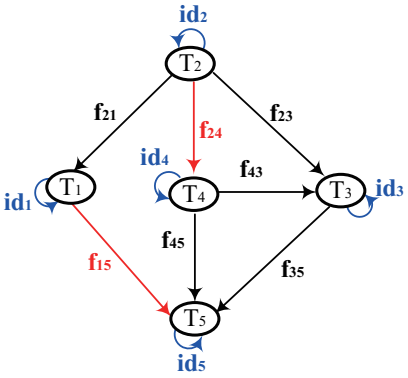


Figure 4. Adding Resource

(seeing Fig. 3) in the above example 1 . The morphisms $f_{16} : T_1 \to T_6$ and $f_{56} : T_5 \to T_6$ means T_6 has lower precedence over its conflicting neighbors T_1 and T_5. The same applies to task T_7.

Rule 2. ***Deleting Task:*** Task that no longer need any resources can be directly removed.

Rule 3. ***Adding Resource:*** The acyclic precedence structure must be preserved between the conflicting tasks, which share the new resource.

Example 3: Two resources Υ_i and Υ_j is added in Fig. 4. Υ_i is shared by T_1 and T_5, and Υ_j is shared by T_2 and T_4.

Rule 4. ***Deleting Resource:*** Resource and its morphism in precedence category $\mathbb{P}$ that no longer need by any tasks can be directly removed.

D. Categorial Model of System

Both of distributed resource allocation system and dynamic resource allocation system consist of a number of components. Each task T_i, containing its local data and code, can be considered as a component. Some tasks can be composed to become subsystem $\Omega = T_i \frown T_j \frown \ldots \frown T_k$. Category theory provides the level of mathematical abstraction to describe software architectures, the composition category $\mathbb{S}$ is defined.

Definition 8. ***Composition Category*** $\mathbb{S} = (\Sigma, \Phi)$**.**

The composition category $\mathbb{S}$ is composed of two collections:

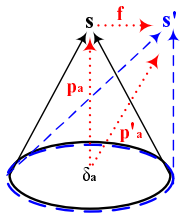


Figure 5. Colimit in Composition Category $\mathbb{S}$

- Σ is a objects collection $\Sigma = \{T_1, T_2, ..., T_n, \Omega_1, \Omega_2, ..., \Omega_m\}$, each object T_i represents a component of a task, and every object Ω_i represents a subsystem, which combined by some components or other subsystems;
- Φ is the collection of morphisms between two different components or subsystems. The *compose morphism* $(\mu_{ij} : T_i \to \Omega_j)$, or $(\nu_{ij} : \Omega_i \to \Omega_j)$ indicates that left component T_i or subsystem Ω_i is a part of the right object Ω_j. For every object T_i or Ω_i in Σ there is an *identity* morphism $id_{T_i} : T_i \to T_i$ or $id_{\Omega_i} : \Omega_i \to \Omega_i$, which means that the component/subsystem is not combined with other component/subsystem.

It's easy to proof the correctness of identify law, composition law and associative law about composition category.

One of the characteristics of universal constructions (such as sum, pushout and colimit) is the ability to capture the collective behaviors of systems of interconnected components. For example, two task component T_i and T_j can be integrated into combined subsystem $T_i \frown T_j$ by using ***sum*** universal operation of category theory; a task component T_i and a subsystem Ω_j can also be integrated into combined subsystem $T_i \frown \Omega_j$ by using ***sum*** universal operation. Hence, we can get a entire-system after using the ***sum*** operation for $n-1$ times. Different systems could be gained through different orders of composition. Each system will be considered as a ***colimit*** of the diagram in the composition category $\mathbb{S}$.

Definition 9. ***Colimit of System.***

Let $\delta : I \to graph(\mathbb{S})$ be a diagram in category $\mathbb{S}$. A colimit of δ is a commutative cocone $p : \delta \to s$, where s is one of the composition of all components, such that, for every other entire-system s' and its commutative cocone $p' : \delta' \to s'$, there is a unique morphism $f : s \to s'$ such that $f \circ p = p'$ (see the Fig. 5). Every entire-system s_i and its commutative cocone $p : \delta \to s_i$ is a colimit of $\delta : I \to graph(\mathbb{S})$.

E. Proof of Properties

We show in this section that our categorical model of distributed and dynamic resource allocation problem has the properties of ***symmetry***, ***safety***, ***liveness*** and ***concurrency***.

Each task executes actions of Critical Section (CS) in limited time after successfully requiring a set of the resources. The condition of performing CS for task T_i, which means that task T_i has higher precedence over all conflicting neighbors $T_j \in T_i.F$, is defined as follows.

Rule 5. ***Accessing CS:***

$$\langle \forall T_j : T_j \in T_i.F : (f_{ij} : T_i \to T_j) \in \mathbb{P} \rangle$$

In the morphism $f_{ij} : T_i \to T_j$, its direction from T_i to T_j doesn't change until T_i complete all actions, therefore, each task satisfy the following rule.

Rule 6. ***Transformation of Precedence:*** The task yields its priority to all its conflicting neighbors only when it finished their operations of CS.

Theorem 1. ***Symmetry:*** All tasks and all resources obey precisely the same rules.

Proof. Every task obeys Rule 5 and Rule 6 for acquiring resources and releasing resources; all tasks and resources can be added or deleted by obeying Rule $1, 2, 3, 4$.

Deadlock, firstly recognized and analyzed in 1968 by E. W. Dijkstra, who termed it as deadly embrace, can cause an indefinite circular wait among some tasks. In resource allocation problem, ***Deadlock*** occurs when two or more tasks in a system are blocked forever, because of requirements that can never be satisfied. The following lemma ensures that our categorical model is free from deadlock.

Lemma 1. ***Free-Deadlock:*** The graph of precedence category is always acyclic.

Proof. The initial graph of precedence category G_0 is a acyclic; Rule 6 ensures a finished task with all of its edges are directed towards to itself, therefore, there is no circle in graph. On the other hand, according to Rule $1, 2, 3, 4$, it is easy to proof that every change (add/delete the tasks/resources) to the graph preserves acyclicity.

Theorem 2. ***Safety:*** Conflicting neighbors never executes the CS simultaneously.

$$T_j \in T_i.F \Rightarrow \neg((f_{ij} : T_i \to T_j) \in \mathbb{P} \wedge (f_{ji} : T_j \to T_i) \in \mathbb{P})$$

Proof. Task T_i and task T_j share a resource, so they are conflict neighbors. If task T_i satisfies Rule 5, therefore task T_i enter to CS. The graph of precedence category is always acyclic (from Lemma 1), so $(f_{ji} : T_j \to T_i) \notin \mathbb{P}$, which means that task T_j doesn't satisfy Rule 5, or, equationally, task j can't access to CS.

Theorem 3. ***Liveness:*** Every task requested resources eventually acquire them and perform its actions.

Proof. A task yields its priority to all its neighbors only when it finished its actions (from Rule 6). Consequently, this task doesn't access to CS once again until that all its conflicting neighbors have finished the execution of actions, equationally, just after the neighbors yield priority to this task.

Theorem 4. ***Concurrency:*** The solution does not deny the possibility of nonconflicting neighbors perform their actions simultaneously.

Proof. A solution to a resource allocation problem involves choosing a minimal set resources for each task such that the minimal sets do not conflict. In this way,

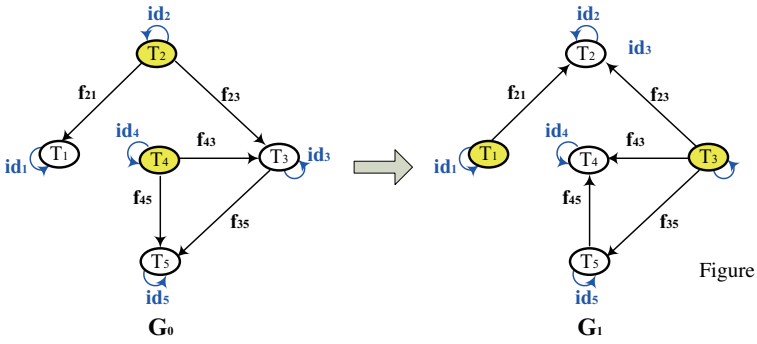


Figure 6. Concurrent Tasks

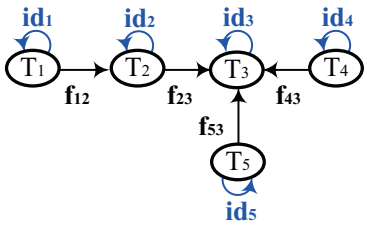


Figure 7. Precedence Category of a Wireless Sensors Network

two or more nonconflicting requested-tasks can perform their actions simultaneously.

In Fig. 6, task T_2 and the task T_4 will can simultaneously enter to CS because they are not neighbors. After the execution, the direction of morphisms associated with task T_2 and task T_2 will change. In the next state of G_1, task T_1 and task T_3 can be perform operations concurrently. There is a a functor $\mathbb{F} : G_0 \to G_1$ between the initial category G_0 and the changed category G_1.

IV. Application Domain

The wireless sensor networks consists of spatially distributed autonomous sensors to monitor physical area (such as enemy intrusion, battlefield surveillance, geo-fencing of gas/oil pipelines, etc) or environmental conditions (such as temperature, sound, pressure, etc). Each **sensor**, which is equipped with a Doppler radar with three **sectors**, is controlled by an independent **agent**[1]. All of the sensor agents must act as a team to cooperatively track the **targets**.

In practice, there is no centralized body to allocate the resources(sensors) and they have to be self organized. In order for a target to be tracked accurately, at least three agents must collaborate - concurrently activating overlapping sectors. If there are multiple targets, an agent may be required by more than two targets. The situation can also be dynamic as targets move through the sensing range, or some targets or sensors my be added or deleted over time. The dynamic feature of the domain makes problems even harder. The categorical models demonstrated in the above section has been applied to the resource allocation among sensors(agents) in distributed and dynamic environment.

Example 4: In a wireless sensor network, there are five sensors $\{\mathsf{S}_1, \mathsf{S}_2, \mathsf{S}_3, \mathsf{S}_4\, \mathsf{S}_5\}$, each sensor S_i has three sectors $\{\mathsf{A}_i^1, \mathsf{A}_i^2, \mathsf{A}_i^3\}$; and five targets $\{\mathsf{T}_1, \mathsf{T}_2, \mathsf{T}_3, \mathsf{T}_4, \mathsf{T}_5\}$.

- Resource set $\Upsilon = \{\mathsf{S}_1, \mathsf{S}_2, \mathsf{S}_3, \mathsf{S}_4\, \mathsf{S}_5\}$, each sensor is considered as a resource;
- Task set $\Gamma = \{\mathsf{T}_1, \mathsf{T}_2, \mathsf{T}_3, \mathsf{T}_4, \mathsf{T}_5\}$, each target is considered as a task;
- Action set $\Lambda = \bigcup_{\mathsf{S}_i \in \Upsilon} Action(\mathsf{S}_i)$, while $Action(\mathsf{S}_i) = \{\mathsf{A}_i^1, \mathsf{A}_i^2, \mathsf{A}_i^3\}$,each sector is considered as a action.

Since a target requires three sensors to track its position, three sectors from three of the five possible sensors are activated. We define a set of corresponding sectors for every targets: $\mathsf{T}_1 \doteq \{\mathsf{A}_1^1, \mathsf{A}_2^1, \mathsf{A}_3^1\}$, $\mathsf{T}_2 \doteq \{\mathsf{A}_2^1, \mathsf{A}_3^2, \mathsf{A}_4^1\}$, $\mathsf{T}_3 \doteq \{\mathsf{A}_1^3, \mathsf{A}_3^2, \mathsf{A}_5^1\}$, $\mathsf{T}_4 \doteq \{\mathsf{A}_1^2, \mathsf{A}_2^2, \mathsf{A}_5^1\}$, $\mathsf{T}_5 \doteq \{\mathsf{A}_1^3, \mathsf{A}_4^2, \mathsf{A}_5^3\}$, where the symbol $\doteq$ means the left task need the right three sectors. T_1 and T_2 are conflicting neighbors because they share a common action A_2^1, the same as other relationships among all targets.

To illustrate the categorical model–*Precedence Category:* $\mathbb{P} = (\Gamma, \Psi)$ defined in this example (seeing Fig. 7), where

- A objects collection $\Gamma = \{\mathsf{T}_1, \mathsf{T}_2, \mathsf{T}_3, \mathsf{T}_4, \mathsf{T}_5\}$,
- A morphisms colletction $\Psi = \{f_{12}, f_{23}, f_{43}, f_{53}, id_1, id_2, id_3, id_4, id_5\}$, while $f_{ij} : \mathsf{T}_i \to \mathsf{T}_j$ mean that target T_i has priority over its conflicting task T_j.

The above *Precedence Category* satisfy all the laws (identify law, associative and transitive law) of category and all the properties defined in Section 3.E. On the other hand, it is easy to extend this formal model to the dynamic sensor network with our strategy.

V. Conclusion

In this paper, we proposed a formalization of resource allocation that is expressive enough to represent both distributed and dynamic aspects of the problem. In order to ensure its correctness, some properties (such as symmetry, safety/non-deadlock, liveness/non-starvation and concurrency) are formally proven. In contrast to other formal models of resource allocation, the perceived benefits of our categorial formalization are as follows: 1) different relationships and interactions can be precisely described by the morphisms of category; 2) traceability and understandability about the solution to resource allocation can be expressed and reasoned using the diagrammatical nature of category theory; 3) system configuration from components can be represented as the universal construction (such as sums, colimits) of category theory. In the future, we will continue extend our formal models to other practical application domain, in addition to wireless sensor networks.

Acknowledgment.

The authors thanks Professor José Luiz Fiadeiro for previous discussion about category theory.

REFERENCES

[1] Pragnesh Jay Modi, Hyuckchul Jung, et al: "A Dynamic Distributed Constraint Satisfaction Approach to Resource Allocation". In: *Principles and Practice of Constraint Programming - CP 2001, Lecture Notes in Computer Science Volume 2239*, 2001, pp. 685-700

[2] P. Danturi, M. Nesterenko, and S. Tixeuil: "Self-stabilizing philosophers with generic conflicts." In: *Proceedings of the 8th International Symposium on Stabilization, Safety, and Security of Distributed Systems*, Dallas, TX, November 2006, pp. 213-230

[3] Xindong You, Jian Wan, Xianghua Xu, Congfeng Jiang, Wei Zhang, Jilin Zhang: "ARAS-M: Automatic Resource Allocation Strategy based on Market Mechanism in Cloud Computing". In: *Journal of Computers*, Vol 6, No 7, 2011, pp.1287-1296

[4] Weifeng Sun, Qiufen Xia, Zichuan Xu, Mingchu Li, Zhenquan Qin: "A Game Theoretic Resource Allocation Model Based on Extended Second Price Sealed Auction in Grid Computing". In: *Journal of Computers*, Vol 7, No 1, Special Issue: Parallel Algorithms, Scheduling and Architectures, 2012, pp.65-75

[5] K. M. Chandy and J. Misra: "The Drinking Philosophers Problem". In: *ACM Transactions on Programming Languages, Systems*, 6(4),1984, pp.632-646

[6] Jingling Yuan, Xin Fu, Tao Li, Minlong Yang: "Intelligent Spatial-based Resource Allocation Algorithms in NoC". In: *Journal of Computers*, Vol 8, No 3, Special Issue: Parallel Computing, 2013, pp.541-549

[7] Urgaonkar R., Kozat U.C., Igarashi, et al: "Dynamic Resource Allocation and Power Management in Virtualized Data Centers". In: *Network Operations and Management Symposium (NOMS), 2010 IEEE*, 2010, pp.479-486

[8] Topaloglu H., Powell W.B: "A Distributed Decision Making Structure for Dynamic Resource Allocation Using Nonlinear Functional Approximations". In: *Operations Research*;53(2):2005, pp.281-297

[9] J.A. Goguen: "A Categorical Manifesto". In: *Math. Struc. comput. Sci.*, 1,1991

[10] M.B. Smyth and G.D. Plotldn: "The category-theoretic Solution of Recursive Domain Equations" In: *SIAM Journal of Computing*,11,1982,pp.761-783

[11] D.J. Lehmann and M.B. Smyth: "Algebraic Specification of Data Types: a Synthetic Approach". In: *Mathematical Systems Theory*, 14,1981,pp.97-139

[12] D.J. Lehmann: "On the Algebra of Order". In: *Journal of Computer and System Sciences*, 21,1980

[13] J.A. Goguen and RiM. Burstall: "Some Fundamental Tools for the Semantics of Computation part 1: Comma Categories, Colimits, Signatures and Theories". In: *Theoretical Computer Science*, 31,1984,pp.175-209

[14] David C. Rine: "A Category Theory For Programming Languages". In: *Mathematical System Theory*, New York. Vol. 7, No. 4,1973, pp.304-317

[15] John C. Reynolds: "Using Category Theory to Design Programming Languages". In: *ESOP 2009, LNCS 5502*, Springer-Verlag, Berlin Heidelberg, 2009

[16] Alexandre Buisse, and Peter Dybjeri: "Towards Formallizing Categorical Models of Type Theory in Type Theory". In: *Electronic Notes in Theoretical Computer Science*, 196,2008, pp.137-151

[17] H.Ehrig, M.Grobe-Rhode, and U.Wolter: "Applications of Category Theory to the Area of Algebraic Specification in Computer Science". In: *Applied Categorical Structures*, 6,1998, pp.1-35

[18] James A. Altuncher and Prakash Panangaden: "A Mechanically Assisted Constructive Proof in Category Theory". In: *Peocedings of the 10th International Conference on Automated Deduction*, (LNCS) Springer-Verlag,1990

[19] D. Kozen, C. Kreitz, E. Richter: "Automating Proofs in Category Theory". In: *Proc. Of IJCAR'06, LNCS, vol.4130*, Springer, 2006, pp.392-407

[20] J. L. Fiadeiro, and T. Maibaum: "A Mathematical Toolbox for Software Architect". In: *Proc. 8th Int. Workshop on Software Sepcification and Design*. IEEE Computer Science Press,Silver Springer, MD,1996, pp.44-55

[21] Xiao Yang, Jinkui Hou, and Jiancheng Wan: "Formal Semantic Meanings of Architecture-Centric Model Mapping". In: *APPT 2007, LNCS 4847*,Springer-Verlag Berlin Heidelberg, 2007, pp.640-649.

[22] J. L. Fiadeiro, and T. Maibaum: "Categorical Semantics of Parallel Program Design". In: *Science of Computer Programming*, 28(2-3),1997, pp.111-138

[23] G. Winskel and M. Nielsen: "Categories in Conurrency". In: *Semantics and Logics of Computation*, Cambridge University Press, 1997, pp. 299-353

[24] R.L. Crole: "Basic Category Theory for Models of Syntax". In: *course notes for Summer School on Generic Programming*, SSGP's 2002

[25] M. Lenisa, J. Power and H. Watanabe: "Category Theory for Operational Semantics". In: *Theoretical Computer Science*, vol. 327, 2004, pp.135-154

[26] Weidman E, Page I, Pervin W: "Explicit dynamic exclusion algorithm". In: *Proc. of the 3rd IEEE Symposium on Parallel*, Distributed Processing, 1991, pp.142-149

[27] K.M. Chandy and J. Misra: "Parallel Program Design: A Foundation". In: Addison-Wesley ,1988

[28] Zhen You, Jinyun Xue, Shi Ying: "Categorical Semantics of a Solution to Distributed Dining Philosophers Problem". In:*FAW 2010, LNCS 6213*, 2010, pp.172-184

[29] Jaap van Oosten: "Basic Category Theory" In*BRICS Lecture notes* LS-95-1, 1995

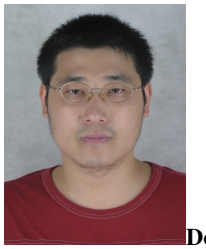
[30] José Luiz Fiadeiro: "Categories for Software Engnieering". In: Springer-Verlag, Berlin Heideberg, 2005

[31] Benjamin C. Pierce. "Basic Category Theory for Computer Scientists". In: The MIT Press, Cambridege, MA, 1991

Zhen You is a Ph.D candidate of computer science from State Key Laboratory of Software Engineering, Wuhan University, Wuhan, China and a Research Assistant from Jiangxi Normal University, Nanchang, China. Her current research interests include concurrent and distributed computing, and formal verification of software.

Jinyun Xue is a Professor from Jiangxi Normal University, Nanchang, China and Ph.D supervisor from Wuhan University, Wuhan, China. He is a senior member of China Computer Federation. His current research interests include formal methods and software automation.

Shi Ying is a Professor and Ph.D supervisor from from State Key Laboratory of Software Engineering, Wuhan University, Wuhan, China. His current research interests include service-oriented software engineering, the method of component-based software engineering.



Dongming Jiang is a Ph.D candidate of computer science from State Key Laboratory of Software Engineering, Wuhan University, Wuhan, China. His current research interests include service-oriented computing.

Qimin Hu is associate professor from Jiangxi Normal University, Nanchang, China. His current research interests include Software components and architecture.