Vol. 3

理论计算机科学

Theoretical Computer Science

上海科学技术文献出版社
Shanghai Scientific and Technological
Literature Publishing House

责任编辑：方金善
封面设计：何永平

理论计算机科学
Vol. 3
《理论计算机科学》编委会　编
*
上海科学技术文献出版社出版发行
（上海市武康路2号　邮政编码200031）
全国新华书店经销
上海科技文献出版社昆山联营厂印刷
*
开本 850×1168　1/32　印张 4　字数 111 000
1998年4月第1版　1998年4月第1次印刷
印数：1—1 000
ISBN 7-5439-1212-0/T·513
定价：9.80元
《科技新书目》455-669

《理论计算机科学》编委会

C. L Liu, University of Illinois
Chitoor V. Ramamoorthy, University of California Berkeley
J. A Robinson, Syrcuse University
Lotfi A. Zadeh, University of California Berkeley

陈火旺中国工程院院士　国防科技大学
Chen Huo Wang　University of National defence Science technology
Academical, The Chinese Academy of engineering
李　祥　贵州大学
Li Xiang　Gui Zhou University
林建祥　北京大学
Ling Jian Xiang　Beijing University
胡国定　南开大学
Hu Guo Ding　Nan Kai University
胡世华院士　中科院软件研究所
Hu Shi Hua　The Chinese Academy of Sciences Institute of Software
Academical
孙永强　上海交通大学
Sun Yong Xiang　Shanghai Jiao Tong University
周巢尘院士　中科院软件研究所
Zhou Chao Chen　The Chinese Academy of Sciences Institute of Software
Academical
朱三元　上海计算机软件技术开发中心
Zhu Sanyuan　Shanghai Development Center of Computer Software Technology

《理论计算机科学》编委会地址
上海成山路 24 弄 8 号 403 室　邮编　200126　电话/传真
58800806
Correspondent Address《The oretical Computer Science》
Eduting Board Shanghai 200126　403Room No8. 24 lone Cheng
Shan Road　Tel/Fax　58800806

理论计算机科学

Theoretical Computer Science

Vol. 3

目　录 Contents

Weak controllability in I/O-CGs, a parallel flow model of computation[①]

Mario Albarran Figueroa
Mathematics and Computer Science Department
San Jose State University, San Jose, CA 95192-0103
albarran @ sjsumcs. sjsu. edu

Abstract

Input/Output Control Graphs (I/O-CG) are presented as a Parallel Flow Model of computation (PFM) suitable for the description and analysis of Discrete Event Systems (DES) which interact with an environment. I/O-CGs, developed by Albarran (1976), stem from Petri Nets (PN), (1972), and are particularly suited for modelling parallel DESs such as computational, manufacturing and robotic systems. I/O-CGs retain the expressive power of PNs to model parallelism while providing explicit modelling of interactions with the environment by means of inputs and outputs. While PNs can produce unbounded state spaces, I/O-CGs always produce finite state spaces while allowing all possible topologies. I/O-CGs, as well as PNs, are essentially non-deterministic and consequently Kalman's determin-

---

① Weak controllability in I/O-CGs, a parallel flow model of computation January 23, 1996

istic controllability is not directly applicable. This paper reports the extension (Albarran, 1976) of the controllability concept to the non-deterministic case. We first introduce the I/O-CG model and formally define it. We show that every I/O-CG can be associated with a transition diagram TD(I/O-CG) which represents its state space. We then group together transitions according to their properties and build paths between states in a TD. A set of definitions which reflect the "degree" of controllability of an I/O-CG is made based on paths which fulfill certain requirements. These paths range from the strictly deterministic case, where all state transitions produce a unique path, to "weaker" forms where state transitions may take different paths for the same input sequence.

Keyword Petri Nets, Weak Controllability

1. The I/O-CG model

An I/O-CG is formed by Control Cells (CC) and Control Operators (CO), similar to places and transitions in PNs respectively. CCs, drawn as squares, can only hold a 1 or a 0 (one or no tokens.) They represent conditions and can be thought of as state variables. COs, drawn as circles, represent events. There are four types of arcs in I/O-CGs:

- An input arc (also called a 1-0 arc) is an arrow from a CC to a CO. CO becomes enabled when CC holds a 1. After a finite, arbitrary amount of time, CO fires changing the 1 into a 0
- An output arc (0-1) joins a CO to a CC. The CO becomes enabled when the CC holds a 0 (no tokens) and after firing CC

holds a 1. Requiring a 0 in the CC ensures no token accumulation

- A read-only arc (1-1) is a double headed arrow joining a CC to a CO. The CO becomes enabled when the CC holds a 1 and after firing the CC keeps the 1
- A zero arc (0-0) is an arrow ending in a small circle that goes from a CC to a CO. The CO operator becomes enabled when the CC holds a 0 and after firing the CC keeps the 0
- A CC with a notch on the left hand side is an Input/Output Control Cell. An I/O-CC gets changed by both, the I/O-CG and its environment and its behavior is similar to that of CCs
- A CO that is enabled with respect to all CCs connected to it is said to be enabled. Once enabled a CO fires changing the values of its connected CC's. When several COs are enabled they all have the same probability of firing but only one, arbitrarily chosen, fires at any one time. Others, if still enabled, follow suit. COs are also called operators while CCs are also called cells.

Figure 1 depicts an I/O-CG that represents the 3 dining philosophers, a simplification of the 5 dining philosophers problem. The philosopher's algorithm is simple :think, grab 2 forks, eat and release 2 forks. Let us separately consider philosopher "Phil 1". Phil 1 is thinking when think 1 holds a 1. Thinking lasts until the philosopher decides to eat. If Phil 1's forks are available (Fork1 and Fork2 both hold 1) then he or she can eat by grabbing them. This set of conditions is enforced by control operator GF1 since it only becomes enabled when think 1 and I/O-CCs Fork 1 and Fork 2 contain a 1. In addition, and different from PNs, CC eat 1 must contain a zero since it is joined to GF 1 by an output (0-1) arc! Output arcs make I/O-CGs different

from PNs by enforcing an interlocking property which prevents the accumulation of tokens in a CC. It is this interlocking property which guarantees the finiteness of the state space of an I/O-CG since any CC can only hold a 0 or a 1.

Once enabled GF1 fires after a finite, arbitrary amount of time resulting in the following changes : think 1, Fork1 and Fork2 contain 0 and eat 1 holds 1. This assignment tells us that Phil 1 is eating and that the forks are being used by him and therefore unavailable.

Notice that Phil 1 is in competition with Phil2 and Phil3 for the use of the forks since these can be taken by either of 2 philosophers in a mutually exclusive fashion. Fork1 can be taken by Phil 1 and Phil 3 (but not by both at the same time!), Fork 2 by Phil 1 and Phil 2 and, finally, Fork 3 by Phil 2 and Phil 3. The arrangement of Fork1, GF 3 and GF 1 is the typical representation of mutual exclusion and ensures that Fork1 is given to either, Phil 1 or Phil 3. When GF 1 and GF 3 are enabled at the same time only one will fire and its occurrence will automatically disable the other since Fork 1 will hold a 0 disabling it.

Initially all philosophers are thinking and all forks are available, Fork1, Fork2, Fork3, think 1, think 2 and think 3 hold 1 and the rest of the cells hold 0. Operators GF 1, GF 2 and GF 3 are all enabled and any one can fire at any one time in a non-deterministic fashion. Once a philosopher eats it releases the forks (when Phil 1 is done eating RF 1 is enabled and fires returning Fork 1 and Fork 2) and the cycle starts over again. Over time one expects the system to be fair to philosophers so that they eat approximately the same number of times. This expectation is true since multiple enabled operators have the same probability of occurrence.

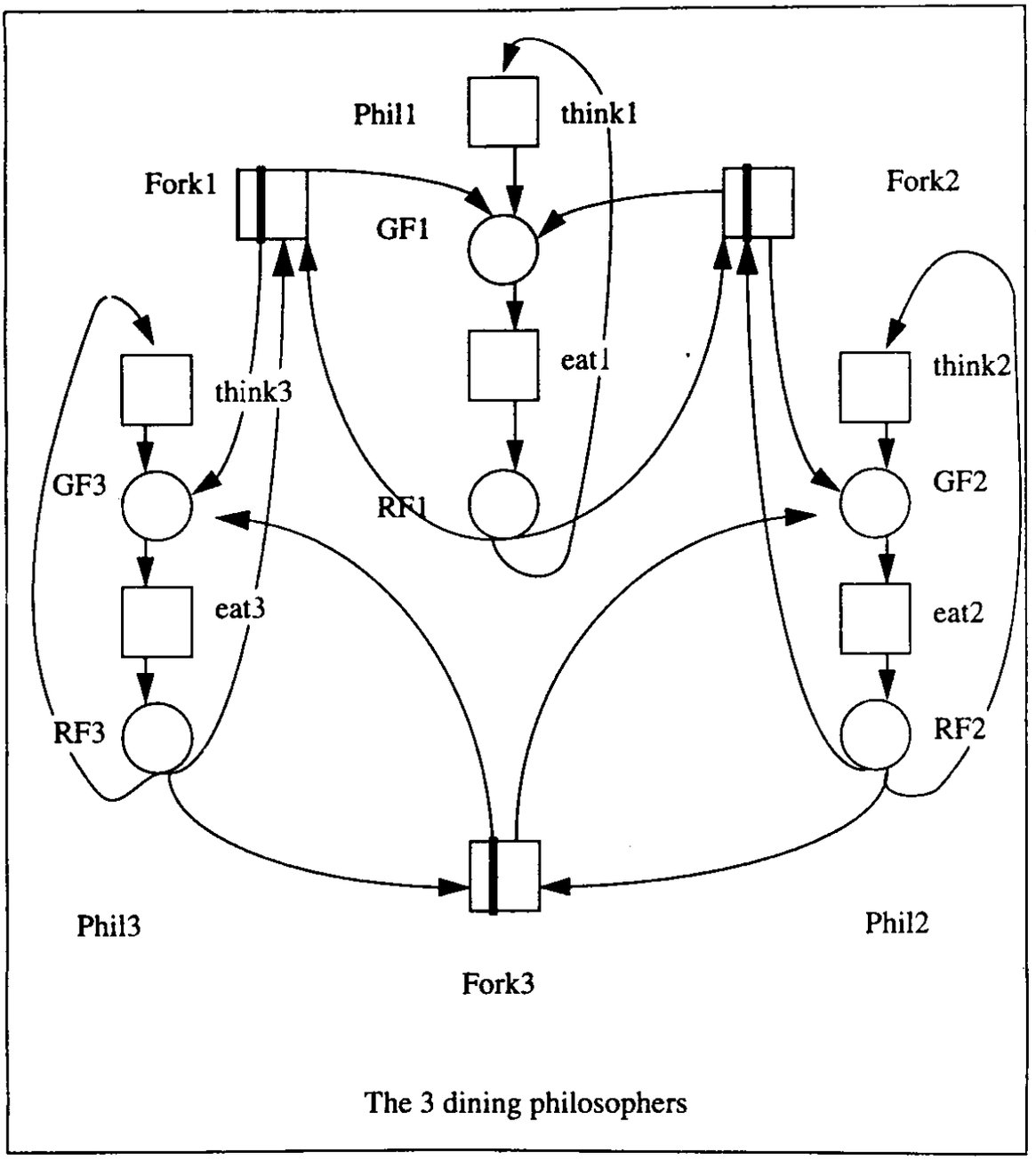

Figure 1

2. I/O-CGs and their Transition diagrams

2.1 Formal definition of an I/O-CG

An I/O-CG is a 9 tuple: I/O-CG = (C, C_E, V, O, A, T_A, t_A, op, cl) where

- C is a set of CCs
- C_E, a non-empty subset of C, is the set of Input/Output CCs
- $V = \{0, 1\}$ is the set of values CCs can hold
- O is a set of control operators
- A is a set of arcs
- $T_A = \{Z, W, I, R\}$ is the set of different types of arcs
- $t_A : A \rightarrow T_A$ assigns each arc a type
- $\mathrm{op} : A \rightarrow O$ connects arcs to operators and
- $\mathrm{cl} : A \rightarrow C$ connects arcs to CCs
- $C_I = C - C_E$ is the set of "internal CCs."

2.1.1 Definition of the state space of I/O-CGs

An internal state Q is an assignment of 0 and 1 values to internal CCs.

- $Q : C_I \rightarrow V$

Internal states, also called states, tell about the state of an I/O-CG at a particular moment. Since CCs can only hold 0 or 1 the maximum number of internal states is 2^n, where n is the cardinality of C.

An environmental assignment assigns 0 and 1 values to I/O CCs.

- $h : C_E \rightarrow V$

A total states is an assignment of 0 and 1 values to all CCs.

- $s : C \rightarrow V$

Total states represent an internal state plus a particular input assignment of the environment to I/O cells. We therefore have two conditions for an operator to become enabled while in state s, internal and environmental enabling.

The internal state space S_I of an I/O-CG is the set of all possible internal states.

$S_I = V^{C_I}$

Since the maximum number of internal states is 2^n the cardinality of S_I is always finite.

2. 1. 2 Properties of COs

Endogenous COs are connected only to internal CCs. Enabling of endogenous COs does not need any input/output from the environment.

Exogenous COs are connected to at least one I/O CC. An endogenous CO becomes "internally enabled" when all CCs contain the values that their connecting arcs require. An exogenous operator becomes "environmentally enabled" when all I/O CCs have the values that their connecting arcs require. To become enabled an exogenous CO needs to be both, internally and environmentally enabled. In figure 1 all operators are exogenous.

2. 1. 3 The initial state and the reachable state space of an I/O-CG

Associated with an I/O-CG there is an initial state S_0 (also called initial marking) which assigns a 0 or a 1 to internal CCs. A marking can be represented as a vector of length N, the number of internal cells. In figure 1 the vector's length is 6: ⟨think1, eat1, think2, eat2, think3, eat3⟩. The initial marking should be ⟨1, 0, 1, 0, 1, 0⟩ since all philosophers are initially thinking. The reachable state space $S_R(S_0)$ of an I/O-CG is the set of all states which can be reached, under all possible inputs, by the I/O-CG when given the initial state S_0.

2. 2 Transition Diagrams

Transition Diagrams (TD) are a general, concise model to represent DESs. A TD is a directed graph where nodes represent states and arcs join states. An arc is drawn from state Q to state R if R results from the occurrence of an event while the DES is in

state Q. Arcs are sometimes labelled with the name of the event or with the input needed. Finite state machines are examples of input driven TDs. Most PFMs of computation rely partly on TDs to draw properties of the systems they model.

Associated with an I/O-CG there is a Transition Diagram TD(I/O-CG) which represents all possible behaviors in a similar way in which a reachability graph is associated with a PNs. Figure 2 represents the diagram for the I/O-CG in figure 1 when given as initial state the ⟨1, 0, 1, 0, 1, 0⟩ vector. A transition from state ⟨1, 0, 1, 0, 1, 0⟩ to state ⟨0, 1, 1, 0, 1, 0⟩ is possible if CCs think 1 and eat1 hold 1 and 0 respectively, making GF1 internally enabled. Since GF 1 is exogenous it needs also to become environmentally enabled by having I/O CCs Fork1 and Fork2 hold 1. By putting ones in Fork1 and Fork2 the environment enables GF1 which produces state ⟨0, 1, 1, 0, 1, 0⟩ after firing and leaves I/O CCs Fork1 and Fork2 holding zeroes. What the next state tells us is that Phil 1 is eating using fork1 and fork2, phil2 and phil3 are waiting for one fork and therefore the only CO enabled is RF1. When RF1 fires the forks are returned.

2. 2. 1 Labels used in Transition Diagrams for I/O-CGs

Let B be the set of I/O-labels

- $B = \{0, 1, d\}C_E$

There are two kinds of I/O-labels associated with an operator p. Input label inlab(p) describes the environmental assignment needed to enable p while output label outlab(p) describes the new values the I/O cells will have after p fires. A d (don't care) in an I/O-label means the operator is not connected to that cell and consequently the cell's value is irrelevant for the operator's enabling. An environmental assignment "covers" an I/O-label if all zeroes and ones in the label are matched by the

values of the I/O cells. Environmental assignment Fork 1 = 1, Fork 2 = 1 and Fork 3 = 0 covers inlab(GF1) = ⟨1, 1, d⟩ in figure 2 making GF1 enabled if the internal state is ⟨1, 0, 1, 0, 1, 0⟩.

Formally, a label for an operator p is a pair of I/O-labels.

- label(p) = ⟨inlab(p), outlab(p)⟩

Arcs in TDs are labelled with a pair: ⟨p, ⟨inlab(p), outlab(p)⟩⟩.

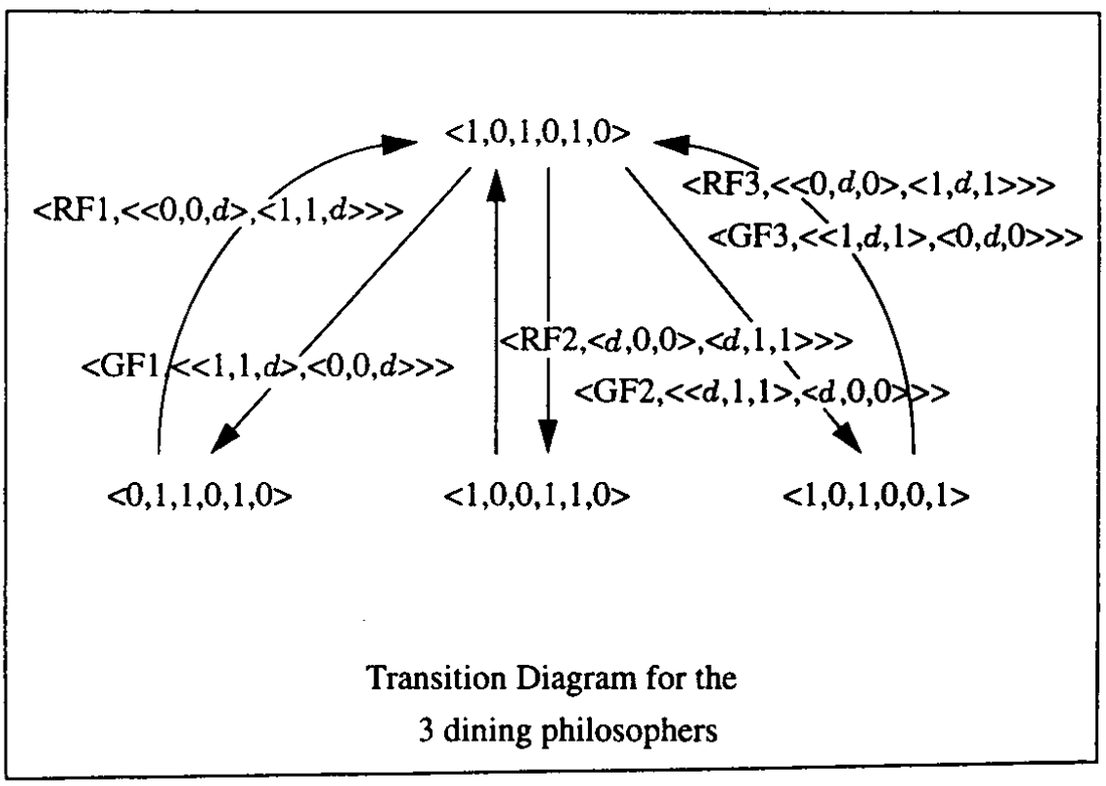

Figure 2

2.2.2 Transition Diagrams for I/O-CGs

A Transition Diagram TD = (S, L, T, ps, ns, lab) for an I/O-CG = (C, C_E, V, O, A, T_A, t_A, op, cl) with an initial state S_0 is obtained by performing the mapping TD(I/O-CG) as follows:

- $S = S_R(S_0)$

Let Q and R be two internal states. L, the set of labels, is a subset of the power set:

- $P(O \times (B \times B))$
- $K(Q, R) = \{\langle p, \text{label } (p)\rangle$ such that R results from Q after p fires$\}$
- $L = \{K(Q, R)$ such that Q and R are in $S\}$

L is the set of labels for all transitions spawned by the I/O-CG from S_0.

T, the set of all transitions, is a subset of $S \times L \times S$

- $T = \{\langle Q, K(Q, R), R\rangle$ such that Q and R are in S

Function ps extracts the previous state from a transition

- ps: $T \rightarrow S$
- ps: $\langle Q, K(Q, R), R\rangle \rightarrow Q$

Function ns extracts the next state from a transition

- ns: $T \rightarrow S$
- ns: $\langle Q, K(Q, R), R\rangle \rightarrow R$

Function lab extracts a transition's label

- lab: $T \rightarrow S$
- lab: $\langle Q, K(Q, R), R\rangle \rightarrow K(Q, R)$

2.2.3 The state space of an I/O-CG

At any time an I/O-CG is in a certain internal state, simply called state, and given a certain input assignment. One or more operators may be enabled. If only one operator is enabled the TD will show only one outgoing arc from that state and we say the transition is a deterministic one. If several operators are enabled the TD will have multiple arcs coming out of that state and we say the transition is non-deterministic since the resulting state is one of a set of possible states. When there are no operators enabled the environment has to provide a different input. A state where no input enables operators is a hung or deadlock state.

A TD represents all possible behaviors of an I/O-CG and can therefore be used as its analysis tool. Knowledge can be ex-