Reliability Analysis of Computer Systems Using Dataflow Graph Models

Krishna M. Kavi
The University of Texas at Arlington
U. Narayan Bhat
Southern Methodist University, Dallas

Key Words—**Dataflow graph, Markov chain, HARP, Decomposition-aggregation, Reliability analysis**

Reader Aids—
Purpose: Tutorial
Special math needed for explanations: Language of graph modeling, probability
Special math needed to use results: Language of graph modeling, probability
Results useful to: Reliability analysts and theoreticians

Summary & Conclusions—**We use dataflow graphs to represent the computational structure, analogous to Petri nets and Turing machines, and have developed a method for analyzing the reliability of computer systems modeled as dataflow graphs. Because of the hierarchical nature of dataflow graphs, systems can be analyzed at several levels of abstraction. Reliabilities of subgraphs can be calculated using either traditional techniques or dataflow approach presented here (recursively). The reliabilities of subgraphs can then be combined leading to decomposition-aggregation approach.**

The time needed for an actor to complete its operation is not included in our analysis of dataflow graphs. Incorporation of the time element compounds the problem and we have not studied it yet.

1. INTRODUCTION

In recent years the dataflow concept has attracted the attention of researchers in the USA, UK, and Japan [12]. Much of the research in dataflow has dealt with the design of dataflow machines and programming languages. The dataflow model can also be used to represent any computation structure including fault-tolerant computers.

Dataflow graph models have been successfully used to simulate computer systems [5, 7, 11]. A formal definition of dataflow graph models is presented in [8]. The chief advantages of dataflow graphs over other models are their compactness and general amenability to direct interpretation. That is, the translation from the conceived system to a dataflow graph is straightforward and, once accomplished, the dataflow graph can be executed using dataflow simulators, (eg, DFDLS [10]) to determine which aspects of the systems are represented. Dataflow graph models are hierarchical; a node in the graph can be a simple node representing a single hardware unit, or a dataflow subgraph representing an entire processor, interconnection network, or a software task. Thus models of computer systems using dataflow graphs can be used to change the level of the model selectively to any degree of detail by expanding nodes into subgraphs, without modifying other parts of the model.

This paper outlines a method for estimating the reliability of dataflow graph models. Computer systems that include parallel processors, and ultrareliable computers can be represented as dataflow graphs; dataflow simulators can be used to study their functionality; and the reliability of the simulated computer can be analyzed using this technique.

This work is supported in part by the NASA-Ames grant NAG 2-273.

Nomenclature

• A *dataflow graph* is a bipartite directed graph. The two vertex classes are *actors* and *links*. Actors represent functions-performed while links are place holders of data values (tokens) as they flow from actor to actor.

• *Uninterpreted dataflow graph*. For the purpose of studying the reliability of a dataflow graph model, the actual meaning of the functions performed by actors and the semantics of the data tokens are not relevent. The presence of tokens at links act as triggering signals to enable actors. Such dataflow graphs are uninterpreted.

Notation

a_i	actor i
ℓ_j	link j
R_i	reliability of actor i
R_{ij}	reliability of the path from a_i to a_j, including reliability of a_i but excluding the reliability of a_j
C_j	reliability of link j
$R^{(k)}$	reliability of path k
$R(G)$	reliability of the dataflow graph

Assumptions

0. The real system can be adequately modeled by a dataflow graph.

1. The performance of an actor is mutually statistically independent of the outcome of other actors.

2. The time needed for an actor to complete its operation is not included in this analysis.

2. DATAFLOW GRAPH MODELS

Dataflow graphs are bipartite directed graphs; the two types of nodes are actors and links. The nodes are interconnected using arcs that can be considered as channels of communication. In the basic form, an actor is enabled for

0018-9529/86/1200-0529$01.00©1986 IEEE

execution only when all the input links to the actor contain tokens and no output links contain tokens [2]. When the actor executes (fires), tokens from the input links are consumed and new tokens are generated on the output links. This mode of sequencing has been extended [8] to permit the execution of actors when only a subset of input links (called input firing semantic set) contains tokens and only a subset of output links (called output firing semantic set) is empty; tokens on the input set are consumed and new tokens are generated on the output set. For different instances of the execution of an actor, the firing semantic sets may be different, thus introducing non-determinacy. Dataflow machines use control tokens and control links to ensure determinate execution of data flow programs [3].

In our formal definition of uninterpreted dataflow graphs [8], the nondeterminate nature of execution is represented by using probability distributions with input and output semantic sets. Based on whether the input firing set and output firing set select: a) one link, b) a proper subset or, c) the entire set of links, the following actor-firing rules are defined.

Conjunctive: All the input links must contain tokens for the actor to fire.

Disjunctive: Only one of the input links must contain a token.

Collective: One or more of the input links may contain tokens for the actor to fire. Collective actors are not considered in this paper.

Selective: When the actor fires, only one of the output links receives a token.

Distributive: When the actor fires all the output links receive tokens.

Figure 1 represents these actor types graphically.

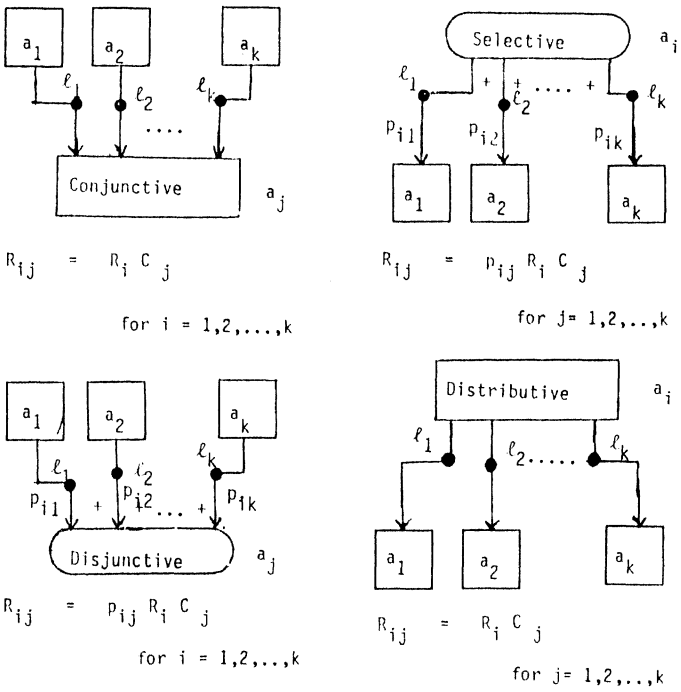


Fig. 1. Reliability Expressions for Dataflow Actors

3. RELIABILITY OF DATAFLOW GRAPHS

Reliability of a dataflow graph can be defined as the probability of successful completion of a sequence of operations to be performed by the actors of the graph. Thus, if this sequence is identified as the path of a particle traversing the graph, then the reliability is the probability of occurrence of a successful path.

Because of the hierarchical nature of dataflow graphs, the reliability of a dataflow graph is determined in two stages.

1. Reliabilities of subgraphs are calculated.
2. Reliabilities of the subgraphs are combined appropriately, based on the topological structure of the graph.

This decomposition-aggregation approach is similar to that of HARP [6] where Petri nets are used. While dataflow graphs are similar to Petri-nets [9], dataflow graphs can also be executed (functional simulation), but Petri nets can only be used in an uninterpreted manner.

In combining subgraph (or actor) reliabilities to get the reliability of the complete graph we have developed reliability expressions (figure 1) based on the actor-firing rules described above (collective actors are not considered). In addition, the following sub-rules are useful.

a. Conjunctive and distributive actors indicate parallel paths all of which are used. When the paths are mutually statistically independent, then the reliability of the graph (or subgraph) consisting of parallel paths is obtained as the product of reliabilities of individual paths.

b. Disjunctive and selective actors result in more than one path, only one of which is used. The reliability of the graph (or subgraph) with such paths is obtained by combining the reliabilities of individual paths using the probabilities of paths as weights.

c. When the paths are not mutually statistically independent, then the dependent structure determines how the reliabilities of individual actors (or subgraphs) are combined to get the reliability of the graph.

d. When redundant paths are simultaneously used, the reliability is obtained using the inclusion-exclusion formula of probability theory [4, pp 99-109].

The following steps can be used to determine the reliability of a dataflow graph.

1. Identify subgraphs.
2. Obtain reliabilities of subgraphs either by traditional methods such as Markov chains or this algorithm recursively.
3. Replace each subgraph by a single actor, thus reducing the graph.
4. Identify distinct paths.
5. Combine actor (subgraph) reliabilities using firing types to determine the reliability for each path.
6. Combine path reliabilities to give the reliability of the entire graph.

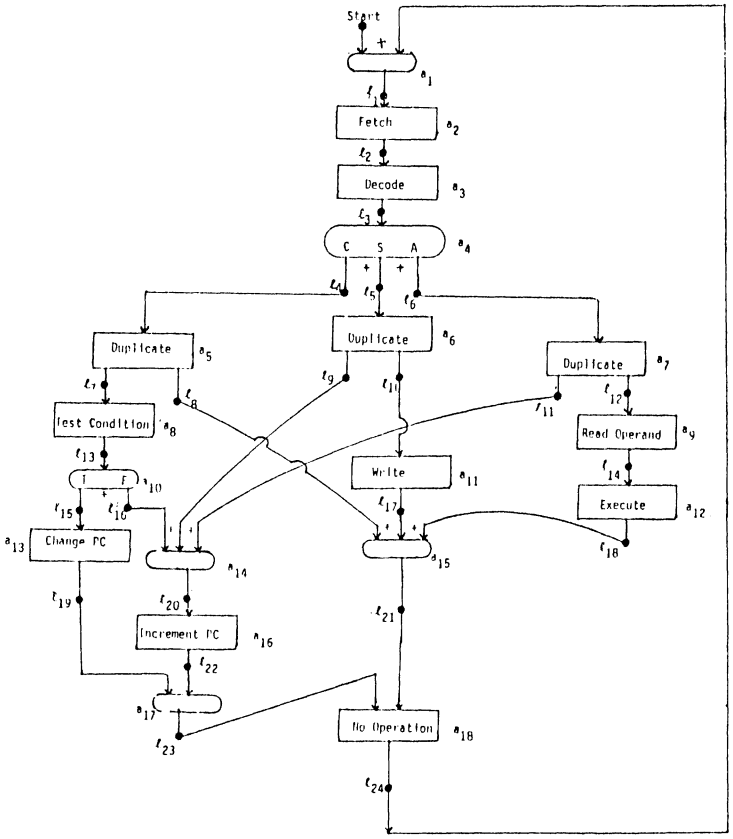
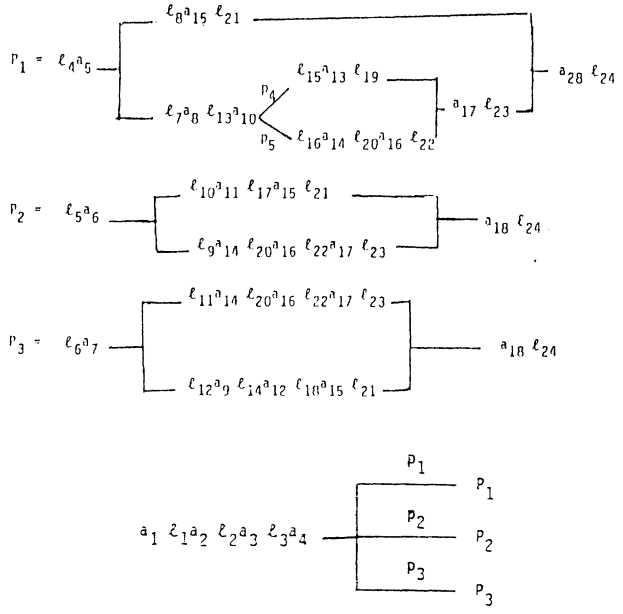


Fig. 2. Dataflow Graph of a Simple Computer System

4. EXAMPLE

Baer [1, p 71] used a Petri-net model to represent the control flow in the execution of an instruction in a single accumulator arithmetic and logic unit. Figure 2 shows a dataflow equivalent of the Petri net given by Baer. The actors are intentionally named by the events in order to facilitate interpretation.

Three distinct paths in the graph are identified:



The probabilities p_1, p_2, p_3 indicate the frequency of Conditional, Store and Arithmetic instructions, respectively (in a typical program); p_4, p_5 are the probabilities that a condition will or will not be satisfied. These are the probabilities defined with the output-firing semantic-sets. The probabilities with input-firing semantic-sets are important for collective actors only.

The reliability of path P_1 is:

$$R^{(1)} = C_4R_5C_8R_{15}C_{21}C_7R_8C_{13}R_{10}(p_4C_{15}R_{13}C_{19} + p_5C_{16}R_{14}C_{20}R_{16}C_{22})R_{17}C_{23}R_{18}C_{24}.$$

The reliabilities $R^{(2)}$ and $R^{(3)}$ of paths P_2 and P_3 can be determined in a similar manner. The system reliability is:

$$R(G) = R_1C_1R_2C_2R_3C_3R_4(p_1R^{(1)} + p_2R^{(2)} + p_3R^{(3)}).$$

REFERENCES

[1] J. L. Baer, *Computer Systems Architecture,* Computer Science Press, 1980.

[2] J. B. Dennis, "First version of data flow procedure language", *Lecture Notes in Computer Science,* vol 19, 1974, Springer-Verlag.

[3] J. B. Dennis, D. P. Misunas, "A preliminary architecture for a basic data flow processor", *Proc. 2nd Annual Symp. Computer Architecture,* (Houston), 1975, pp 126-132.

[4] W. Feller, *An Introduction to Probability and Its Applications,* John Wiley & Sons, 1968.

[5] J. L. Gaudiot, M. D. Ercegovac, "Performance analysis of dataflow computers with variable resolution actors", *Proc. 4th Conf. Distributed Comp. Systems* San Francisco, 1984 May, pp 2-9.

[6] R. Geist, K. S. Trivedi, J. B. Dugan, M. Smotherman, "Design of the Hybrid Automated Reliability Predictor", *Proc. IEEE/AIAA Digital Avionics Systems Conf.,* 1983 Oct, pp 1.16.5.1 - 16.5.8.

[7] K. M. Kavi, "Data flow modeling techniques", *Proc. IASTED Intl. Conf. Modeling and Simulation,* Orlando, FL, 1983 Nov, pp 1-4.

[8] K. M. Kavi, B. P. Buckles, U. N. Bhat, "A formal definition of dataflow graph models", *IEEE Trans. Computers,* 1986 Nov.

[9] K. M. Kavi, B. P. Buckles, U. N. Bhat, "Isomorphisms between Petri nets and dataflow graphs", Tech. Rept., #CSE TR 85-001.*

[10] K. M. Kavi, R. M. Boyd, S. R. Amble, "DFDLS: A dataflow language based on graphical structure and a dataflow simulator", Tech. Rept., #CSE 86-002.*

[11] V. P. Srini, J. F. Asenjo, "Analysis of Cray 1S architecture", *Proc. 10-th Computer Architecture Symp.* (Stockholm), 1983 June, pp 194-206.

[12] P. C. Treleaven, D. R. Brownbridge, and R. P. Hopkins, "Data driven and demand driven computer architecture", *ACM Computing Surveys,* 1982 Mar, pp 93-143.

* Available from: Science Engineering; University of Texas; Arlington, Texas 76019-0015 USA.

AUTHORS

Dr. U. Narayan Bhat; Dept. of Statistics; Southern Methodist University; Dallas, Texas 75275 USA.

U. Nayayan Bhat is a professor in the department of Statistics at SMU. Bhat published extensively on queuing theory and its applications, and reliability of computer systems. He authored the book, *Elements of Applied Stochastic Processes,* (2nd edition, John Wiley & Sons). He served as an associate editor for *Operations Research, Management*

(*continued on page 528*)