TELKOMNIKA Indonesian Journal of Electrical Engineering
Vol.12, No.5, May 2014, pp. 3266 ~ 3271
DOI: http://dx.doi.org/10.11591/telkomnika.v12i5.4945

Test Method for Process Deadlock Based on Graph Grammars

Yi Wang*[1,2], Han Ding[1], Fan Yang[1]
[1]School of Mathematical and Computer Sciences, Hubei University of Arts and Science,
Xiang Yang, 441053, China, No.296 of Longzhong Road
[2]Institute of Intelligence Science and Technology, Hohai University,
Nanjing, 210098, P.R. China
*Corresponding author, e-mail: dhnats@163.com

Abstract

This paper proposes a test method for process deadlock based on graph grammars through constructing process resource diagram. Through using the construction rules, it can construct and judge the validity of the process resource diagram. Through using the test rules, it can test if there is the deadlock in the process. The method is a graphical approach; it is simple and intuitive with strong operability.

Keywords*: process, deadlock, test, graph grammars*

1. Introduction

The basic characteristics of operating system are concurrent and sharing. A system allows multiple processes to execute concurrently, and shares the software and hardware resources. In order to maximize the use of the computer system resources, operating system should adopt the strategy of dynamic allocation for the system resources. Using this strategy, however, when the application number is bigger than the entry number for a certain types of resource, if the allocation is improper, it may occur that waiting for resources between the processes and could not move forward, make the processes block each other. In fact, the application for the sources between the different processes may get some meet according with a certain order; this is likely to cause two or more processes mutual blockade. Each process “catches” some resources, which the other processes are waiting for. The result is that whichever process also can not get the all resources, and all of these processes can not continue to run.

Traditional method of testing the process deadlock descripts by text, but the text is long and the way of thinking is not clear. This paper, based on the principle of graph grammars, uses the construction rules construct and judge the validity of the process resource diagram. Through using the test rules, it can test if there is the deadlock in the process. The method is a graphical approach; it is simple and intuitive with strong operability.

2. Graph Grammars

Graph grammar [1-5] is used to define and analyze the figure. The figure is abstracted as a two-dimensional object with nodes and edges. Graph grammar involves operations include derivation and reduction. In the application, they were also used in a variety fields [6-9]. Basic concepts and operations are introduced simply as follows:

Definition 1 G= (V, E, LV, LE, S, T) is a figure, among them:

V is a set of nodes of a graph G, composed with terminal node set VT and non-terminal node set VN;

E is a set of edges of a graph G;

LV is a set of nodes labels of a graph G;

LE is a set of edges labels of a graph G;

Received October 15, 2013; Revised November 25, 2013; Accepted December 12, 2013

S: E →V, T: E →V are two functions that gives the start node and the end node of each edges of a graph G;

Definition 2: A production of a graph grammar is a rule likes gl : = gr. gl and gr are two graphs, called as the left and the right of the production separately.

Definition 3: Host graph, expressed as ghost, is a graph that will be transformed using productions. In additional, a subgraph of a graph that is isomorphic with gl is expressed as glhost, which will be replaced as a graph that isomorphic with gr when a graph is transformed.

Definition 4: From ghost, remove all the nodes and the edges of a glhost and the edges which endpoint on glhost will get a graph, called as Residual Graph, marked as gresidual.

Definition 5: grhost is a graph, which is isomorphic with gr, and replaces glhost of a host graph.

The productions is the basis of the transformation for a host graph, but in order to complete the transformation, only production is not enough, the corresponding rules also should explain how to embed grhost into gresidual, the rules called embedding rules.

Definition 6: A graph grammar gg is a triple (A, P, E), among them, the A is the initial graph of the graph grammar, the P is the set of the productions, the E is the embedding rules.

Definition 7: Set gg = (A , P, E) is a graph grammar, L (gg) = { G| A => G∧ there is not non-terminal in G} is a language set of a graph grammar.

For a graph grammar, there are two operations: one is derivation, the other is parsing. The prior means that from the host graph to find out a subgraph that isomorphic with the left graph of the production and replaced with a graph that isomorphic with the right graph of the production; the latter means that from a graph to find out a subgraph that isomorphic with the right graph of the production and replaced with a graph that isomorphic with the right graph of the production. The graphs that derived from the host graph called as the language of the graph grammar. Through parsing, if a graph can arrive to the initial character, the graph is called as a language of the graph grammar.

3. Construction for Process Resource Diagram Based on Graph Grammar

Definition 1: A process resource diagram (PRD) can be represented as PRD=(P, R, Er, Ed), among them:

P is a set of processes;

R is a set of resources;

Er is a set of edges according to a process requests for resources;

Ed is a set of edges according to a resource assigned to a process.

For example, Figure 1 is a PRD,

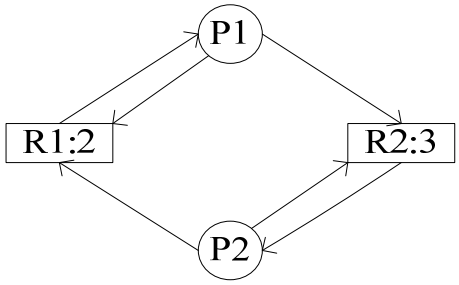

Figure 1. A PRD

Among them, P= {p1, p2}, R= {R1, R2}, Er= {<p1, R1>, <p1, R2>, <p2, R1>, <p2, R2>}, Ed= {<R1, p1>, <R2, p2>}.

< pi, Rj> presents a process pi applies a resource Rj;

< Rj, pi> presents a resource Rj has been allocated to a process pi.

Definition 2: A reasonable process resource diagrams is a PRD in that the processes have not been deadlocked in a state. A reasonable process resource diagram should satisfy two conditions:

The allocation for a resources Rj can not exceed the total;

The sum of the number of any process pi applies for a resource Rj and the number of the resource Rj allocate to the processes is no more than the total of the resource Rj.

Definition 3: A complete schematic diagram is a PRD that contains only an isolated node.

Based on the basic principle of graph grammars, through the rules and using the derivation of the graph grammar it can construct the process resource diagram. Design rules are as Figure 2:

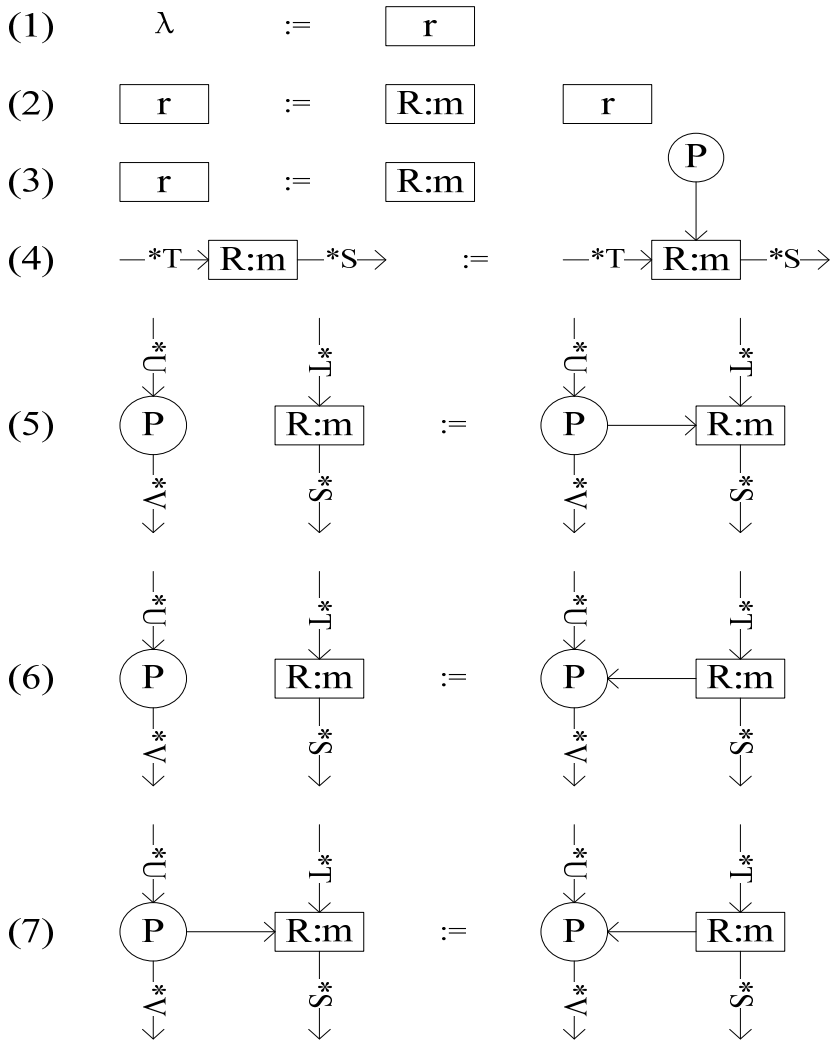

Figure 2. A Set of the Rule to Construct the Process Resource Diagram

Among them, (P) means a process; [R:m] means a kind of resource, m is the number of the resource; [r] is a non-terminal node that of a resource.

Rule 1 means that it begin to draw a resource, that is to say to draw a process resource diagram;

Rule 2 means starting from a non-terminal node of a resource and bring in a terminal node;

Rule 3 means that all the non-terminal nodes of the resources changed into the terminal nodes and the resources had been brought end;

Rule 4 bring in a process and the process puts forward an application to a resource;

Rule 5 means that an existing process puts forward an application to a resource;

Rule 6 distributes a resource to a process;

Rule 7 means that an application that a process puts forward to a resource been changed to the allocation that a resource to a process.

Figure 3 is the construction process of the Figure 1,

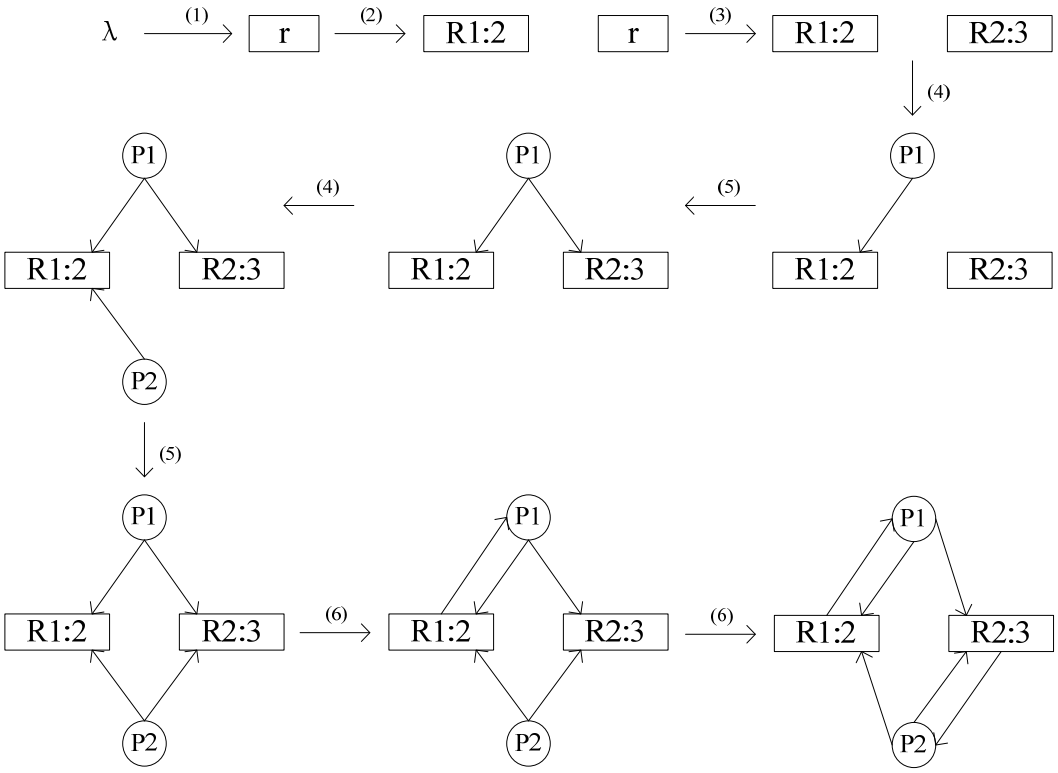

Figure 3. The Construction Process of the Figure 1

4. Test for Process Deadlock Based on Graph Grammar

Theorem: The system is a deadlock state, if and only if its process resource diagram can not change to a complete schematic diagram.

Based on the theorem, we can judge if it have a deadlock in a system.

Step 1: In the process resource diagram to look for if it have a process node that the sum of the application edges and the allocation edges less than the total of the resource. If it is yes then it turn to step2, else it turn to step3;

Step 2: For the process node that satisfy the condition, we can modify the application edges to the allocation edges, and this makes the process node be an isolated node; Turn to step 1;

Step 3: If it is a complete schematic diagram, the system does not exist the deadlock.

Based on the basic principle of graph grammars, through the rules and using the parsing of the graph grammar it can construct the process of the test for process deadlock. Design rules are as Figure 4. Among them:

Rule 1 puts an application edge change to an allocation edge for a process node;

Rule 2 deletes all the allocation edges of a resource allocate to a process;

Rule 3 put two isolated nodes parse to a isolated node;

Rule 4 put a terminal node resource parse to a non-terminal node resource;

Rule 5 put a terminal node resource and a non-terminal node resource parse to a non-terminal node resource;

Rule 6 put an isolated resource and an isolated process parse to the initial character.

From the process of the test for process deadlock of the Figure 1, we can see that the Figure 1 can be parsed to the initial graph: λ, so the Figure 1 does not exist the deadlock.

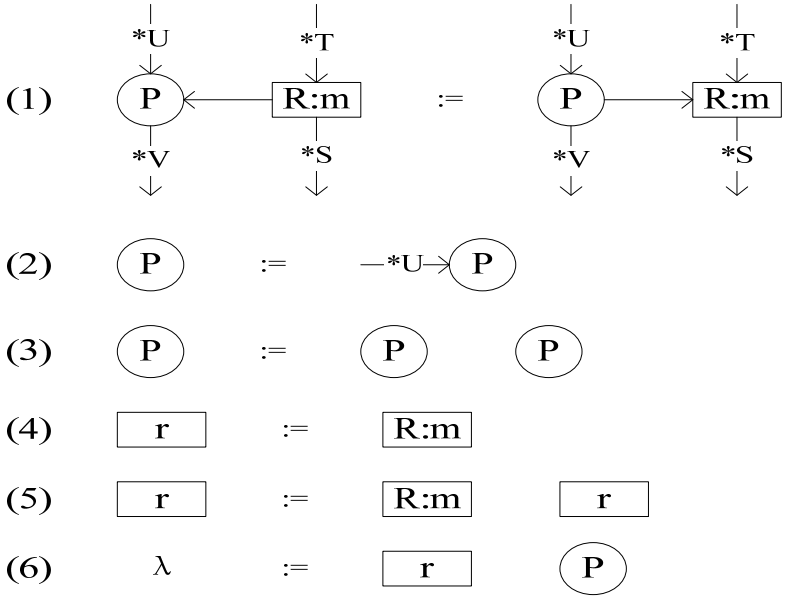

Figure 4. A set of the Rule to Construct the Test for Process Deadlock

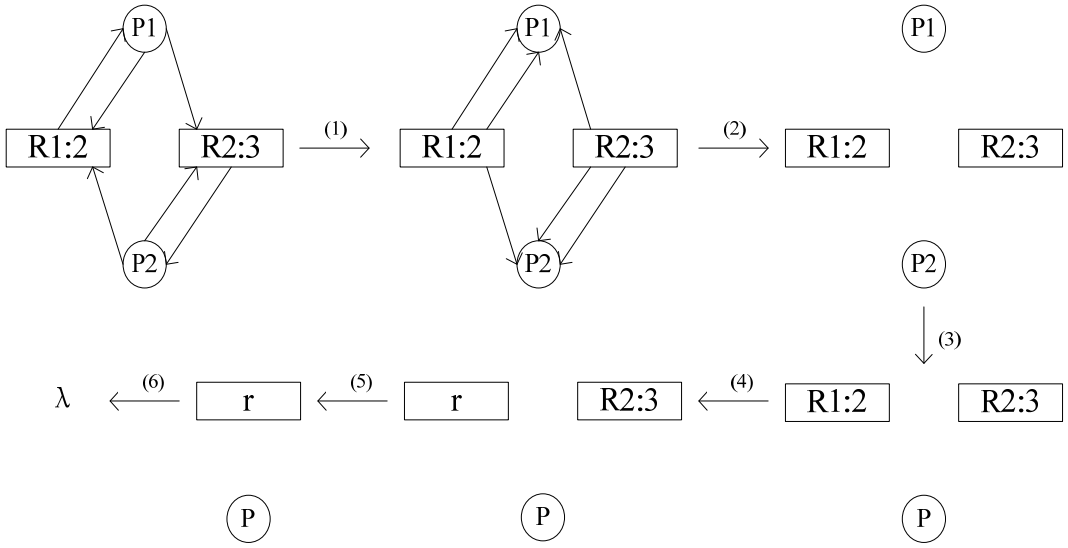

Figure 5. The Test for Process Deadlock of the Figure 1

5. Summarize and Look Forward

This paper proposes a test method for process deadlock based on graph grammars through constructing process resource diagram. Through using the construction rules, it can construct and judge the validity of the process resource diagram. Through using the test rules, it can test if there is the deadlock in the process. The method is a graphical approach; it is simple and intuitive with strong operability.

Acknowledgements

This paper is supported by Hubei Provincial Department of Education Research Project (B2013101)

References

[1] Rozenberg G. Handbook of Graph Grammars and Computing by Graph ransformation. Singapore: world scientific Publishing. 1997; 1 1.
[2] Ehrig H, Engels G, Kreowski H2 J, et al. *Handbook of Graph Grammars and Computing by Graph Transformation: Applications, languages and Tools*. Singapore :World scientific Publish2ing ,1999; 2.
[3] Ehrig H, Kreowski H2 J, Montanari U, et al. *Handbook of Graph Grammars and Computing by Graph Transformations: Concurrency, parallelism, and Dist ribution*. Singapore: World scientific Publishing. 1999; 3.
[4] Drewes F, Hoffmann B, J anssens D, et al. Adaptive Star Grammar. ICGT. 2006: 77-91.
[5] Lara J, Bardohl R, Ehrig H, et al. Attributed Graph Transformation with Node Type Inheritance. *Theoretical Computer Science.* 2007; 376(3): 1392163.
[6] Pfaltz J. Web Grammars and Picture Description. *Computer Graphics and Image Processing*. 1972; 1(1): 193-220.
[7] Bunke H. Attributed Programmed Graph Grammars and Their Application to Schematic Diagram Interpretation. *IEEE Pattern Analysis and Machine Intelligence*. 1982; 4(6): 574-582.
[8] Fahmy H, Blostein D. *A Graph2Grammar Programming Style for Recognition of Music Notation. Machine Vision and Applications, to appear 1992 Preliminary result*. Proc. First International Conference on Document Analysis & Recognition. St. Malo, France. 1991: 70-78.
[9] Dolado J, Torrealdea F. Formal Manipulation of Forrester Diagrams by Graph Grammars. *IEEE System, Man and Cybernetics.* 1988; 18(6): 981-996.