ISSN 0254–2757

QI QUÆSTIONES INFORMATICÆ

Volume 6 • Number 1

May 1988

The official journal of the Computer Society of South Africa and of the South African Institute of Computer Scientists

Die amptelike vaktydskrif van die Rekenaarvereeniging van Suid-Afrika en van die Suid-Afrikaanse Instituut van Rekenaarwetenskaplikes

QUÆSTIONES INFORMATICÆ

The official journal of the Computer Society of South Africa and of the South African Institute of Computer Scientists

Die amptelike vaktydskrif van die Rekenaarvereniging van Suid-Africa en van die Suid-Afrikaanse Instituut van Rekenaarwetenskaplikes

Subscriptions

The annual subscription is

	SA	US	UK
Individuals	R20	$ 7	£ 5
Institutions	R30	$14	£10

to be sent to:
Computer Society of South Africa
Box 1714 Halfway House 1685

Wirth / Hoare / Dijkstra / Gries /
Bach

Kürchen

Kowalski

Lizette

Guest Editorial.

Quæstiones Informaticæ is prepared by the Computer Science Department of the University of the Witwatersrand and printed by Printed Matter, for the Computer Society of South Africa and the South African Institute of Computer Scientists.

Editorial

Volume six of QI heralds several changes. The most visible is the change in format. The black on red cover has been changed to a more readable blue on white, but we have retained the style of the old cover, for the sake of continuity. The papers are now set in a tighter format, using double columns, which will enable more papers to be published for the same cost.

For authors, the most significant change is that as from Volume 6 Number 2 (the next issue), a charge will be made for typesetting. The charge is quite modest – R20 per page – and will enable us to keep up the high standards that we have become used to with QI. It is worth recording that the alternative to this suggestion was that authors should present camera-ready typescript, as is done for *Quæstiones Mathematicæ*. Given that document preparation and electronic typesetting is one of the areas of computer science that we can feel proud of, it seemed right that our journal should use the most modern techniques available. Fortunately, the two controlling bodies, the CSSA and SAICS, eventually agreed to our proposal and the result is the professional journal you have in front of you now.

Supporters of QI may be interested in a few statistics that I compiled when I took over the editorship from Gerrit Wiechers in April this year. In the past two years (June 1985 to June 1988), 73 papers have been received. Of these 39 (53%) have appeared, 19 have been rejected or withdrawn (26%) and 15 (21%) are either with authors for changes or with referees. If we look at the complete picture for Volumes 4 and 5, we find the following:

Volume	Issues	Papers	Pages	Ave. pages per paper
5	3	27*	220	7.7
4	3	21	136	6.4

Although this issue contains one very long paper of 18 pages, the future policy of QI will be to restrict papers to 6 or 7 printed pages, and prospective authors are asked to bear this in mind when submitting papers.

For the future, we are hoping to move towards more special issues. Many of the papers being published at the moment were presented at the 4th SA Computer Symposium in 1987. Instead of continuing the policy of allowing such papers to be accepted by QI without further refereeing, we are hoping to negotiate with Conference organisers to produce special issues of QI. Thus the proceedings would *ab initio* be typeset by QI and all the papers would be in a single issue. Given the competitive charges of QI, there will be financial gains for both parties in such an arrangement.

As this is my first editorial, it is fitting that it should close with a tribute to the previous QI team. My predecessor as editor was Gerrit Wiechers. Gerrit took over the editorship in 1980 and served the journal well over the years. With his leadership, the number and quality of the papers increased to its present healthy state. I must also extend a big thank you to Conrad Mueller and the University of the Witwatersrand who pioneered desk top publishing of QI in August 1985, using the IBM mainframe and its laser writer. Without Conrad's diligence and the excellent facilities provided by the Wits Computer Centre and subsequently the Computer Science Department, QI would easily have degenerated into a second-rate magazine. Quintin Gee, also of the Wits Computer Science Department, has taken over from Conrad and has raised the production quality of QI to new heights, as this issue testifies.

I look forward to your help and support in the future. Long live QI!

Judy M Bishop
Editor
June 1988

Protection Graph Rewriting Grammars and the Take/Grant Security Model

S H von Solms and D P de Villiers
Department of Computer Science, Rand Afrikaans University, PO Box 524, Johannesburg, 2000

Abstract

The operations in the Take/Grant Protection Model are formalised using theory and results from the discipline of formal languages. A Protection Graph Rewriting Grammar is defined, which generates protection graphs consistent with the restrictions inherent in the Take/Grant Model.
Keywords: *Take/Grant Model, protection graph, formal grammar, rewriting system*

Received July 1987, Accepted July 1987

1. Introduction

The goal of this paper is to define a graph rewriting grammar to simulate the operations in the Take/Grant security model [2]. This grammar will have certain context conditions applicable to every production in the grammar. These context conditions will allow the grammar to simulate the conditions inherent in the different operations allowed in the Take/Grant model.

The grammar will rewrite one protection graph [1] by another, the rewriting process being controlled by the context conditions of the productions.

For a discussion of the Take/Grant model, the reader is referred to [1], [2].

2. Protection Graphs

2.1 Definition
A protection graph is a directed, loop-free, edge labelled, two colour graph
$P = (V,R)$
where $P = V \cup O$, $S \cap O = \varnothing$, is called the set of nodes, with S the set of subject and O the set of object nodes.

Edges are labelled by nonempty subsets of a finite set of labels
$R = \{r_1, ..., r_n\} \cup \{t,g\}$, called *rights*.
R contains two distinguished elements t and g. We say P is a protection graph over V/R.

2.2 Definition
For any protection graph $P = (V,R)$, let $\psi(V/R) =$
$$\{\underset{x}{\bullet} \xrightarrow{R_1} \underset{y}{\bullet} \mid x,y \in V, R_1 \subseteq R\},$$
i.e. $\psi(V/R)$ is the set of all protection graphs over V/R consisting of only two nodes.

An element $D \in \psi(V/R)$ is called a *limited* protection graph.

x is called the initiator and y the receiver of the limited protection graph
$$D = \underset{x}{\bullet} \xrightarrow{R_1} \underset{y}{\bullet} \text{ denoted by } i(D) \text{ and } r(D)$$
respectively.

3. Protection Graph Rewriting Grammars

3.1 Definition
A protection graph rewriting grammar (PGR-grammar), is a system
$G = (V,\Sigma,P,I,R)$ where
V is a finite non-empty set of nodes, $V = S \cup O$
S is called the subjects, and O is called the objects.
Σ is a finite non-empty subset of V, called the set of terminal nodes.
P is a set of productions described below.
I is the initial protection graph (axiom).
R is a finite non-empty set of edge labels, with the two distinguished labels t and g in R.

P consists of 4 types of productions:
3.1.1 Growth production
A growth production can rewrite (replace) a node by a limited protection graph, i.e. it can extend (let grow) an existing protection graph. (Add a new node to the protection graph). Growth productions have the following form:
$(x, D, (X_1 ; X_2), (Y_1, Y_2))$
where $x \in V$, D is a limited protection graph over V/R with $x = i(D)$
$X_1, X_2 \subseteq \Sigma$
Y_1, Y_2 are limited protection graphs over V/R
$r(D) \in X_2$.

A production of this kind is applied in the following way to a protection graph H containing x:
$$\text{Suppose } D = \underset{x}{\bullet} \xrightarrow{R_1} \underset{y}{\bullet}, R_1 \subseteq R.$$

Add y as a node to H, with a new directed edge, labelled R_1 between x and y, if the following conditions hold:

(a) All elements of X_1 appear somewhere in H,
(b) No elements of X_1 appear anywhere in H,
(c) All elements of Y_1 appear as subgraphs somewhere in H,
(d) No elements of Y_2 appear as subgraphs anywhere in H.

X_1 and X_2 are known as the permitting/ forbidding node contexts respectively, and Y_1 and Y_2 as the permiting/ forbidding edge contexts.

From the description above, it is clear that this type of production allows the protection graph under consideration to grow, i.e. add new nodes.

Note y ε Y_2 (from the definition of this kind of production). This requirement prevents the addition of y if y already appears in the relevant graph.

3.1.2 Edge generation productions

An edge generation production can generate (insert) an edge between two existing nodes in a protection graph.

Edge generation productions are of the form:
$(x, y, R_1, (X_1, X_2), (Y_1, Y_2))$, where
$(x, y \,\varepsilon\, V, R_1 \subseteq R, X_1, X_2, Y_1, Y_2)$ are defined and used as in 3.1.1.

The effect of such a production is to insert the edge labelled by R_1 between nodes x and y if the context conditions are satisfied.

Note that x, y ε X_1, i.e. both x and y must appear in the protection graph under consideration. Further we demand that

$\underset{x}{\bullet}\xrightarrow{R_i}\underset{y}{\bullet}$, $\forall R_i \subseteq R$ be in Y_2, i.e. there may not (already) exist an edge between x and y.

3.1.3 Edge removal productions

An edge removal production can remove an existing edge between two nodes.

These productions have the form:
$(x, y\ (X_1, X_2), (Y_1, Y_2))$
with x, y, X_1, X_2, Y_1, Y_2 defined as in 3.1.1.

The effect of such a production is to remove the edge between nodes x and y if the context conditions are satisfied.

Note that $\underset{x}{\bullet}\xrightarrow{R_1}\underset{y}{\bullet}$, must be in Y_1 i.e. there must be an existing edge, labelled R_1, between x and y.

3.1.4 Edge label update productions

These productions can change the label of an existing edge.

Their form are:
$(x, y, R_1, R_2, (X_1, X_2), (Y_1, Y_2))$,
where x, y, X_1, X_2, Y_1, Y_2 are defined as in 3.1.1, and $R_1, R_2 \subseteq R$.

The effect of such a production is to change the label of the edge between x and y from R_1 to R_2.

Note that $\underset{x}{\bullet}\xrightarrow{R_1}\underset{y}{\bullet}$, must be in Y_1 i.e. there must be an existing edge, labelled R_1, between x and y.

3.2 Definition (Informal)

The language generated by a protection graph rewriting grammar
$G = (V, \Sigma, P, I, R)$
is the set of all protection graphs over Σ/R that can be generated from the axiom I using productions from P.

3.3 Definition

If $\Sigma = V$, we call G an extended protection graph rewriting grammar.

In the rest of this paper we will assume, without stating it every time, that $\Sigma = V$.

We will also not distinguish explicitly between object and subject nodes.

4. Note

We have now generated different protection graphs from the axiom. This generation was strictly controlled by the context conditions of the different productions.

The "definition" of a PGR-grammar is based on [3].

5. Simulating the Take/Grant Model with PGR-Grammars

In this paragraph we will take the four rewriting rules in the Take/Grant model [2], and simulate them with PGR-productions. We will discuss each rewriting rule separately, primarily to show how the context conditions in the PGR-production "controls" the specific Take/Grant rewriting rule.

5.1 The Take-rule

Let x, y and z be nodes in a protection graph PG_1, such that x is a subject. Let there be an edge from x to y, labelled R_1, such that "t" ε R_1, and an edge from y to z labelled R_2.

Let $R_3 \subseteq R_2$.

Then the "take"-rule defines a new protection graph PG_2 by adding an edge to PG_1 from x to z.

Graphically

$$\begin{tikzcd} \underset{x}{\bullet} \arrow[r, "R_1"'] & \underset{y}{\bullet} \arrow[r, "R_2"'] & \underset{z}{\bullet} & \Rightarrow & \underset{x}{\bullet} \arrow[r, "R_1"'] \arrow[rr, bend left, "R_3"] & \underset{y}{\bullet} \arrow[r, "R_2"'] & \underset{z}{\bullet} \end{tikzcd}$$

We can simulate this using an edge generation production as discussed in 3.1.2. Consider the following edge generation production:

5.1.1 $(x, z, R_3, (y\ ;\ ),$

$$(\underset{x}{\bullet} \xrightarrow{R_1=\{t\}\cup\bar{R}_1} \underset{y}{\bullet}\ ,\ \underset{y}{\bullet} \xrightarrow{R_2} \underset{z}{\bullet}\ ;\ \underset{x}{\bullet} \xrightarrow{R_4} \underset{z}{\bullet}))$$

$\forall\ \varepsilon\ S$, (remember $V = S \cup 0$), $R_3 \subseteq R_2$, $\forall\ R_4 \subseteq R$.

Production 5.1.1 states:
The node y must appear in PG_1
The limited protection graphs

$$(\underset{x}{\bullet} \xrightarrow{R_1=\{t\}\cup\bar{R}_1} \underset{y}{\bullet} \text{ and } \underset{y}{\bullet} \xrightarrow{R_2} \underset{z}{\bullet}$$

must appear in PG_1, and the limited protection graph

$\underset{x}{\bullet} \xrightarrow{R_4} \underset{z}{\bullet}$ may not appear for any $R_4 \subseteq R$, i.e.

there may be no existing edge between x and z in PG_1. So PG_2 is generated from PG_1 using production 5.1.1.

Note that 5.1.1 is actually a shorthand notation for a whole set of productions. Every member of the set can be written down explicitly, making the semantic requirements viz

$x\ \varepsilon\ S$, $R_3 \subseteq R_2$, $\forall\ R_4 \subseteq R$ unnecessary.

Using this set of productions we can therefore mechanically implement the Take-rule by some automata.

5.2 The Grant-rule

Let x, y and z be distinct nodes in protection graph PG_1 such that x is a subject. Let there be an edge from x to y labelled R_1, such that "g" $\varepsilon\ R_1$, and an edge from x to z labelled R_2. Let $R_3 \subseteq R_2$. The "grant"-rule defines a new protection graph PG_2 by adding an edge from y to z labelled R_3. Graphically

$$\begin{tikzcd} \underset{x}{\bullet} \arrow[r, "R_1=\{g\}\cup\bar{R}_1"'] \arrow[rr, bend left, "R_2"] & \underset{y}{\bullet} & \underset{z}{\bullet} \arrow[r, phantom, "\Rightarrow"] & \bullet \arrow[r, "R_1"'] \arrow[rr, bend left, "R_2"] & \underset{y}{\bullet} \arrow[r, "R_3"'] & \underset{z}{\bullet} \end{tikzcd}$$

Again we can simulate this using an edge generation production. Consider the following edge generation production:

5.2.1 $(y, z, R_3,\ (x\ ;\ ),$

$$(\underset{x}{\bullet} \xrightarrow{R_2} \underset{z}{\bullet}\ ,\ \underset{x}{\bullet} \xrightarrow{R_1=\{g\}\cup\bar{R}_1} \underset{y}{\bullet}\ ;\ \underset{y}{\bullet} \xrightarrow{R_4} \underset{z}{\bullet}))$$

for $x\ \varepsilon\ S$, $R_3 \subseteq R_2$, $\forall\ R_4 \subseteq R$.

The explanation of this production follows the same lines as 5.1.1.

5.3 The Create-rule

Let x be any subject node in a protection graph PG_1, and let R_1 be a non-empty subset of R. The "create"-rule defines a new protection graph PG_2 by adding a new node n to PG_1 with an edge from x to n labelled R_1. Graphically

$$\underset{x}{\bullet} \Rightarrow \underset{x}{\bullet} \xrightarrow{R_1} \underset{n}{\bullet}$$

We simulate this using a growth production as described in 3.1.1. Consider the following growth production:

$$(x, \underset{x}{\bullet} \xrightarrow{R_1} \underset{n}{\bullet}; (\ ;\ \{n\})\ (\ ;\ )$$

for $x\ \varepsilon\ S$, $R_1 \subseteq R$.

The forbidding node context checks that n does not already appear in PG_1, i.e. two identical nodes cannot appear in PG_2.

5.4 The Remove-rule

Let x and y be distinct nodes in a protection graph PG_1 such that x is a subject. Let there be an edge from x to y labelled R_1, and let R_2 be any subset of rights. The "remove"-rule defines a new protection graph PG_2 by deleting the R_2 rights from R_1.

Graphically,

$$\underset{x}{\bullet} \xrightarrow{R_1} \underset{y}{\bullet} \Rightarrow \underset{x}{\bullet} \xrightarrow{R_1 - R_2} \underset{y}{\bullet}$$

We can simulate this using an edge label update production as described in 3.1.4. Consider the following production: (Let $R_3 = R_1 - R_2$).

$$(x, y, R_1, R_3\ (\ ;\ )\ (\underset{x}{\bullet} \xrightarrow{R_1} \underset{y}{\bullet};)$$

The permitting edge context requires that an edge between x and y, labelled R_1, must exist in PG_1.

If $R_3 = \varnothing$, the edge can be physically removed using an edge removal production as discussed in 3.1.3.

6. Conclusion and Further Research

From the approach taken in this paper, it seems possible to maintain a secure environment by using the PGR-productions to enforce the restrictions and context conditions inherent in any security policy. The decision making process, i.e. deciding which rights may be given to whom, and who may access whom, can be "hard-coded" into the system using the productions. The PGR-grammar can then automatically decide, by checking the situation of the present protection graph, and by applying applicable productions, where and when access is allowed.

The next step is to try to model the Send/Receive security model using the same principles.

References

[1] J. Biskup, [1984], Some Variants of the Take-Grant Protection Model, *Information Processing Letters*, **19**, 151-156.
[2] L. Snyder, [1981], Formal Models of Capability-Based Protection Systems, *IEEE Trans on Computers* **C-30** (3), 172-181.
[3] S.H. von Solms, [1984], Node-Label-Controlled Graph Grammars with Context Conditions, *International Journal of Computer Mathematics*, **13**.

This paper first appeared in the Proceedings of the 4th South African Computer Symposium.

NOTES FOR CONTRIBUTORS

The purpose of the journal will be to publish original papers in any field of computing. Papers submitted may be research articles, review articles and exploratory articles of general interest to readers of the journal. The preferred languages of the journal will be the congress languages of IFIP although papers in other languages will not be precluded.

Manuscripts should be submitted in triplicate to:

Professor J M Bishop
Department of Computer Science
University of the Witwatersrand
Johannesburg
Wits
2050

Form of manuscript

Manuscripts should be in double-space typing on one side only of sheets of A4 size with wide margins.

The first page should include the article title (which should be brief), the author's name and affiliation and address. Each paper must be accompanied by an abstract less than 200 words which will be printed at the beginning of the paper, together with an appropriate key word list and a list of relevant Computing Review Categories.

Manuscripts may be provided on disc using any Apple Macintosh package or in ASCII format.

For authors wishing to provide camera-ready copy, a page specification is freely available on request from the Editor.

Tables and figures

Tables and figures should not be included in the text, although tables and figures should be referred to in the printed text. Tables should be typed on separate sheets and should be numbered consecutively and titled.

Figures should also be supplied on separate sheets, and each should be clearly identified on the back in pencil with the authors name and figure number. Original line drawings (not photocopies) should be submitted and should include all the relevant details. Photographs as illustrations should be avoided if possible. If this cannot be avoided, glossy bromide prints are required.

Symbols

Mathematical and other symbols may be either handwritten or typewritten. Greek letters and unusual symbols should be identified in the margin. Distinction should be made between capital and lower case letters; between the letter O and zero; between the letter I, the number one and prime; between K and kappa.

References

References should be listed at the end of the manuscript in alphabetic order of the author's name, and cited in the text in square brackets. Journal references should be arranged thus:

[1] E. Ashcroft and Z. Manna, [1972], The Translation of 'GOTO' Programs to 'WHILE' programs, *Proceedings of IFIP Congress 71,* North-Holland, Amsterdam, 250-255.

[2] C. Bohm and G. Jacopini, [1966], Flow Diagrams, Turing Machines and Languages with only Two Formation Rules, *Comm. ACM,* **9**, 366-371.

[3] S. Ginsburg, [1966], *Mathematical Theory of Context-free Languages,* McGraw Hill, New York.

Proofs

Proofs will be sent to the author to ensure that the papers have been correctly typeset and *not* for the addition of new material or major amendment to the texts. Excessive alterations may be disallowed. Corrected proofs must be returned to the production manager within three days to minimise the risk of the author's contribution having to be held over to a later issue.

Only original papers will be accepted, and copyright in published papers will be vested in the publisher.

Letters

A section of "Letters to the Editor" (each limited to about 500 words) will provide a forum for discussion of recent problems.