A Matrix-based Method for Analysing Stochastic Process Algebras*

J.T. BRADLEY
University of Durham, UK

N.J. DAVIES
University of Bristol, UK

Abstract

This paper demonstrates how three stochastic process algebras can be mapped on to a generally-distributed stochastic transition system. We demonstrate an aggregation technique on these stochastic transition systems and show how this can be implemented as a matrix-analysis method for finding steady-state distributions. We verify that the time complexity of the algorithm is a considerable improvement upon a previous method and discuss how the technique can be used to generate partial steady-state distributions for SPA systems.

Keywords

Stochastic process algebras, semi-Markov processes, Markov renewal processes, partial evaluation of steady-state distributions, algorithm complexity, matrix-based analysis.

1 Introduction

In this paper, we describe a matrix analytic method for producing steady-state distributions in a state-wise fashion from semi-Markov processes. We present a mapping from some popular SPAs to their equivalent semi-Markov processes. We then demonstrate how the SMPs can be aggregated to a normal form, in which an individual steady-state probability can be calculated.

The reduction to the normal form is specifically targeted for analysis of an individual state, so each reduction will only yield a single state of the steady-state distribution. Thus the method can be used to look at specific states for which the distribution is required. If, say, a reward vector operating on a distribution is particularly sparse, then only the portion of the steady-state distribution for which the reward vector is non-zero will need to be calculated.

*Supported by NATS, a division of the CAA, and QWiD, EPSRC project GR/N01491.

The stochastic processes and systems which can be analysed by this method are ones which have a finite expansion as semi-Markov processes and thus include continuous-time Markov chains, Markovian process algebras, generally-distributed pre-emptive restart process algebras, and phase-type and Coxian distributed pre-emptive resume process algebras. Therefore in figures 1–3 we provide a mapping from PEPA [5], MTIPP [4] and a generally-distributed SPA [1] to their underlying stochastic transition system (which is our algebraic notation for a semi-Markov process).

In the remainder of this paper we will briefly summarise stochastic transition systems and the normal form reduction technique. We will then go on to describe the matrix method which describes this reduction and discuss issues of algorithm complexity.

2 Stochastic Transition Systems

2.1 Stochastic Transition Systems and Semi-Markov Processes

A semi-Markov process (or Markov renewal process) is defined as $\{Z(t), t \geq 0\}$, where $Z(t)$ represents the state of a process at time t. From a state i, there is a probability p_{ij}, $i, j \geq 0$ that the system will enter state j next with an arbitrarily distributed delay, $F_{ij}(t)$. The Markovian label comes from the fact that on entering each state there is no memory of any previous state.

Traditionally, steady-state distributions of semi-Markov processes are found from the embedded Markov chain which has P, elements p_{ij}, as its transition matrix. Thus $\mathbf{q} = P\mathbf{q}$ is solved in the usual way and the steady-state distribution of the original semi-Markov process is obtained: $\pi_i = \mu_i q_i / \sum_j \mu_j q_j$, where μ_i is the expected sojourn time in the ith state.

Clearly this still requires the calculation of the entire steady-state distribution of the embedded Markov process. What we achieve in this paper is the direct calculation of individual steady-state probabilities of the semi-Markov process which are functions solely of the sojourn time in a state and the return time back to that state (section 2.3). The return distribution, a useful stochastic quantity in itself, is calculated using a smaller matrix and thus less computation than for the entire steady-state distribution (section 3). This gives us the possibility of calculating individual steady-state probabilities in less time than it would take to calculate the whole distribution using traditional methods.

We use a stochastic transition system as a basic algebra to describe a semi-Markov process: a state S is defined as:

$$S ::= \{X\}.S \;\big|\; [p]S \oplus [1-p]S \;\big|\; A \tag{1}$$

where X represents a continuous positive random variable and p, $0 < p < 1$, is a probability and A is a constant label. The $\oplus$ operator represents a probabilistic branch in the transition system.

2.2 Constructing Mappings to SPAs

We use a set of probabilistic mapping rules to turn various traditional stochastic process algebras into their equivalent stochastic transition systems. The mapping rules operate in a similar fashion to the functional semantics of [8] with the difference that they add the possibility of probabilistic operation. So in figure 1, we interpret the competitive choice rule as: if $P \xrightarrow{(a,\lambda)} P'$ and $Q \xrightarrow{(b,\lambda)} Q'$ then we can infer that with probability $\lambda/(\lambda+\mu)$, $P+Q \xrightarrow{(a,\lambda+\mu)} P'$ or with probability $\mu/(\lambda+\mu)$, $P+Q \xrightarrow{(a,\lambda+\mu)} Q'$. This embodies the competitive nature of the race condition and can be mapped directly onto the probabilistic branch construct of a stochastic transition system.

Figure 1 shows the mapping rules for PEPA [5], figure 2 contains the mapping for MTIPP [4] and figure 3 describes a generally-distributed stochastic process algebra with pre-emptive restart [1]. As can be seen from the restart rule, the fact that a pre-empted process gets restarted means that a history of residual distributions does not need to be kept by the algebra. The Last-to-Finish and First-to-Finish synchronisation models referred to in figure 3 are the subject of [2] and [1].

2.3 Stochastic Normal Forms

2.3.1 Introduction

In the context of the rest of the paper a Cox-Miller Normal Form from [1] is a 2-state stochastic transition system such that: $A = \{X\}.B$ and $B = \{Y\}.A$. It is useful because it has a steady-state distribution that is easily calculated [1, 9]: $\pi_A = \mathbb{E}(X)/(\mathbb{E}(X)+\mathbb{E}(Y))$ and $\pi_B = \mathbb{E}(Y)/(\mathbb{E}(X)+\mathbb{E}(Y))$.

2.3.2 Motivation

In this section, we show that general stochastic transition systems can be reduced to a normal form, using stochastic aggregation. The simplest of these normal forms is the Cox-Miller Normal Form as described above. In the next section we will show how the other normal form cases can themselves be solved using the Cox-Miller result.

To understand the process of aggregation to the normal form, we should explain the stochastic motivation. This, in turn, will explain the structure of the normal forms.

Given any chosen state, whose steady-state probability we wish to find, we have to maintain the stochastic characteristics of that state within the system. In a stochastic transition system, the system remains in a given state for a random amount of time, i.e. the *sojourn time* in the state. The exact distribution of this amount of time is defined by the distributions on the leaving transitions.

Prefix

$$\frac{}{(a,\lambda).P \xrightarrow{(a,\lambda)} P}$$

Competitive Choice

$$\frac{P \xrightarrow{(a,\lambda)} P' \quad Q \xrightarrow{(b,\mu)} Q'}{\begin{array}{l}\frac{\lambda}{\lambda+\mu} : P+Q \xrightarrow{(a,\lambda+\mu)} P' \\ \frac{\mu}{\lambda+\mu} : P+Q \xrightarrow{(b,\lambda+\mu)} Q'\end{array}}$$

Cooperation (Blocking)

$$\frac{P \xrightarrow{(a,\lambda)} P' \quad Q \xrightarrow{(b,\mu)} Q'}{P \underset{S}{\bowtie} Q \xrightarrow{(b,\mu)} P \underset{S}{\bowtie} Q'} \quad a \in S, b \notin S$$

Cooperation (Passive)

$$\frac{P \xrightarrow{(a,\lambda)} P' \quad Q \xrightarrow{(a,\top)} Q'}{P \underset{S}{\bowtie} Q \xrightarrow{(a,\lambda)} P' \underset{S}{\bowtie} Q'} \quad a \in S$$

Cooperation (Pre-emptive Restart/Resume)

$$\frac{P \xrightarrow{(a,\lambda)} P' \quad Q \xrightarrow{(b,\mu)} Q'}{\begin{array}{l}\frac{\lambda}{\lambda+\mu} : P \underset{S}{\bowtie} Q \xrightarrow{(a,\lambda+\mu)} P' \underset{S}{\bowtie} Q \\ \frac{\mu}{\lambda+\mu} : P \underset{S}{\bowtie} Q \xrightarrow{(b,\lambda+\mu)} P \underset{S}{\bowtie} Q'\end{array}} \quad a, b \notin S$$

Cooperation (Active)

$$\frac{P \xrightarrow{(a,\lambda)} P' \quad Q \xrightarrow{(a,\mu)} Q'}{P \underset{S}{\bowtie} Q \xrightarrow{(a,R)} P' \underset{S}{\bowtie} Q'} \quad a \in S$$

$$\text{where } R = \frac{\lambda}{r_a(P)} \frac{\mu}{r_a(Q)} \min(r_a(P), r_a(Q))$$

Fig. 1. Probabilistic mapping for PEPA to a stochastic transition system. (The function $r_a(\cdot)$ is defined in [5]. The definitions of Constant and Hiding are not shown for reasons of space, but are in any case identical to their functional definitions, also in [5].)

Prefix

$$\frac{}{(a,\lambda).P \xrightarrow{(a,\lambda)} P}$$

Competitive Choice

$$\frac{P \xrightarrow{(a,\lambda)} P' \quad Q \xrightarrow{(b,\mu)} Q'}{\begin{array}{l}\frac{\lambda}{\lambda+\mu} : P+Q \xrightarrow{(a,\lambda+\mu)} P' \\ \frac{\mu}{\lambda+\mu} : P+Q \xrightarrow{(b,\lambda+\mu)} Q'\end{array}}$$

Synchronisation (Blocking)

$$\frac{P \xrightarrow{(a,\lambda)} P' \quad Q \xrightarrow{(b,\mu)} Q'}{P \,||_S\, Q \xrightarrow{(b,\mu)} P \,||_S\, Q'} \quad a \in S, b \notin S$$

Synchronisation (Pre-emptive Restart/Resume)

$$\frac{P \xrightarrow{(a,\lambda)} P' \quad Q \xrightarrow{(b,\mu)} Q'}{\begin{array}{l}\frac{\lambda}{\lambda+\mu} : P \,||_S\, Q \xrightarrow{(a,\lambda+\mu)} P' \,||_S\, Q \\ \frac{\mu}{\lambda+\mu} : P \,||_S\, Q \xrightarrow{(b,\lambda+\mu)} P \,||_S\, Q'\end{array}} \quad a,b \notin S$$

Synchronisation (Scaling factor)

$$\frac{P \xrightarrow{(a,\lambda)} P' \quad Q \xrightarrow{(a,\mu)} Q'}{P \,||_S\, Q \xrightarrow{(a,\lambda\mu)} P' \,||_S\, Q'} \quad a \in S$$

Fig. 2. Probabilistic mapping for MTIPP to a stochastic transition system.

Prefix

$$\frac{}{(a,X).P \xrightarrow{(a,X)} P}$$

Competitive Choice

$$\frac{P \xrightarrow{(a,X)} P' \quad Q \xrightarrow{(b,Y)} Q'}{\begin{array}{l}\mathbb{P}(X < Y) : P + Q \xrightarrow{(a,X|X<Y)} P' \\ \mathbb{P}(Y < X) : P + Q \xrightarrow{(b,Y|Y<X)} Q'\end{array}}$$

Synchronisation (Blocking)

$$\frac{P \xrightarrow{(a,X)} P' \quad Q \xrightarrow{(b,Y)} Q'}{P \,||_S\, Q \xrightarrow{(b,Y)} P \,||_S\, Q'} \quad a \in S, b \notin S$$

Synchronisation (Pre-emptive Restart)

$$\frac{P \xrightarrow{(a,X)} P' \quad Q \xrightarrow{(b,Y)} Q'}{\begin{array}{l}\mathbb{P}(X < Y) : P \,||_S\, Q \xrightarrow{(a,X|X<Y)} P' \,||_S\, Q \\ \mathbb{P}(Y < X) : P \,||_S\, Q \xrightarrow{(b,Y|Y<X)} P \,||_S\, Q'\end{array}} \quad a, b \notin S$$

Synchronisation (Last-to-Finish)

$$\frac{P \xrightarrow{(a,X)} P' \quad Q \xrightarrow{(a,Y)} Q'}{P \,||_S\, Q \xrightarrow{(a,\max(X,Y))} P' \,||_S\, Q'} \quad a \in S$$

Synchronisation (First-to-Finish)

$$\frac{P \xrightarrow{(a,X)} P' \quad Q \xrightarrow{(a,Y)} Q'}{P \,||_S\, Q \xrightarrow{(a,\min(X,Y))} P' \,||_S\, Q'} \quad a \in S$$

Fig. 3. Probabilistic mapping for a generally-distributed stochastic process algebra with pre-emptive restart.

Therefore, a state in such a system is defined by its out-edges and preserving the stochastic characteristic of the state means preserving its out-edges. Under this restriction we would like to be able to simplify the remainder of the system as much as possible. The next section shows that for each out-edge from a state, the rest of the system can be stochastically aggregated to a single state with a single return edge. So we create a class of systems with the same number of two-state cycles as there are out-edges from the state that we were originally carefully "aggregating around", shown in figure 4.

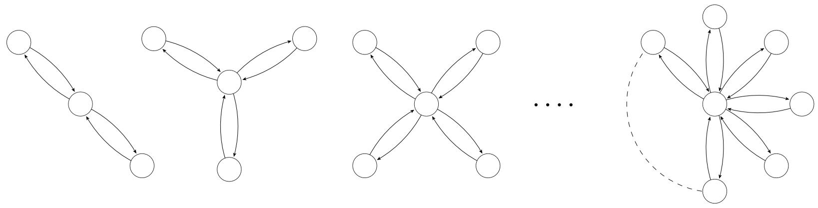

Fig. 4. The set of normal forms for a state with finite branching degree.

In doing this, we have fulfilled our wish to preserve the stochastic nature of the state, since each out-edge still exists explicitly in the reduced system. We have also preserved the context of the state within the original system, since the "remainder of the system" behaviour is modelled in aggregated form by each of the return edges in the cycles. These edges effectively represent the aggregated system behaviour given that a particular out-edge was traversed.

2.3.3 Solution of Normal Forms

Cox-Miller Normal Form only occurs when aggregating around a state with a single out-transition. We use this fact to solve the remaining normal forms. The normal forms of figure 4 each consist of a central vertex of out-degree $n \geq 2$. All the adjacent vertices to the central state, i.e. the next ones out along the out-edges, have out-degree one.

This is the key. Using the stochastic aggregation technique again, we can take each of these "out-degree one" vertices in turn and aggregate to obtain a Cox-Miller Normal Form. It is a simple matter now to find the stationary probabilities for each of the degree one vertices and thus obtain the central vertex stationary probability by summing and subtracting from 1.

More specifically, let us label the central vertex with out-degree $n \geq 2$ as v. Label the n adjacent vertices to v, v_i for $1 \leq i \leq n$. For each v_i, aggregate to obtain a Cox-Miller Normal Form and solve for the stationary probability π_{v_i}. Now $\pi_v = 1 - \sum_{i=1}^{n} \pi_{v_i}$.

3 A Matrix Algorithm for Creating Normal Forms of Stochastic Transition Systems

3.1 Introduction

A graph-based proof of reduction of stochastic transition systems to their normal forms was presented in [10], however the exponential complexity of the technique prohibits its use as a practical tool. This graph-based method used the aggregation rules presented in [1] and thus was identical to the method used to reduce flow graphs in [6] (it should be noted that this flow graph method was not applied to reduce systems to a normal form, as here; it was solely a method of aggregation). In this section, we demonstrate a polynomial-time matrix algorithm which can reduce an arbitrary stochastic transition system to its normal form.

In the algorithm presented, Laplace transforms are used to represent the distributions so that we can take advantage of the convolution result for random variables, namely: $L_{X+Y}(z)$ can be constructed from $L_X(z)L_Y(z)$.

3.2 Solving An Example

Using an example we will illustrate the main points of the algorithm before restating it more generally at the end of the section.

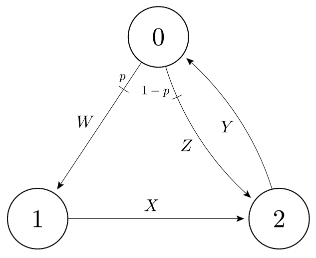

Fig. 5. An example stochastic transition system described by transition matrix, T.

For the stochastic process example in figure 5, we construct a transition matrix, T:

$$T = \begin{pmatrix} 0 & pL_W(z) & (1-p)L_Z(z) \\ 0 & 0 & L_X(z) \\ L_Y(z) & 0 & 0 \end{pmatrix} \tag{2}$$

where t_{ij} represents the Laplace distribution of the direct transition from state i to state j. In a similar fashion to transition matrices in Markov chains, this allows us to obtain possibly useful quantities such as T^n which represents the n-transition matrix and $(e_iT) \cdot \begin{pmatrix} 1 & 1 & 1 \end{pmatrix}$ which represents the departure distribution from a given state, i. (e_i is the unit vector, consisting of the elements $(e_i)_j = \delta_{ij}$, where $\delta_{ij} = 1$ if $i = j$ and 0 otherwise.)

However, what we require is a matrix M with elements m_{ij}, which represents the passage-time distribution [3] from a state i to a state j. This passage-time distribution represents the return distributions of the normal forms described earlier. By finding the return distributions between each of the states, we can easily construct the normal forms, and in doing so demonstrate that the normal forms exist for those states.

The purpose of this section is to demonstrate a way of constructing M from T. We will proceed by constructing an individual column of M for this example and then in the next section easily generalise to the whole matrix.

For state 1 in the example system of figure 5, the column of the return distribution matrix M that we are interested in is m_{j1}, for $0 \leq j \leq 2$. This column vector represents the return distributions to state 1 from states 0, 1 and 2. These distributions can then be used directly in the return transitions of the appropriate normal form; in this case, the normal form for state 1 (figure 6) would only have a single out-edge and thus only a single return edge. The reason that the other distributions need to be generated is that they may be referenced by the equation which yields the desired return distribution, as will be seen.

In generating the return distributions, we need to prune the transition system to remove the out-edges from the state under consideration, and make it an absorbing state for the system. To achieve this, we need to modify T appropriately by replacing the 1-row of T by the unit vector e_1, $\begin{pmatrix} 0 & 1 & 0 \end{pmatrix}$.

We notice that a return distribution from state 2 to state 1, say, can be constructed by convolving a Y transition with the return distribution from 0 to 1. In a similar way, using this modified system, we can construct a recursive relation between all the return distributions, which is terminated by the absorbing state 1:

$$\begin{pmatrix} m_{01} \\ m_{11} \\ m_{21} \end{pmatrix} = \begin{pmatrix} 0 & pL_W(z) & (1-p)L_Z(z) \\ 0 & 1 & 0 \\ L_Y(z) & 0 & 0 \end{pmatrix} \begin{pmatrix} m_{01} \\ 1 \\ m_{21} \end{pmatrix} \tag{3}$$

giving two equations in two unknowns, m_{01} and m_{21}, and a third equation, $m_{11} = 1$, which makes state 1 an absorbing state. The quantities m_{01} and m_{21} can now be solved using standard linear algebra techniques to obtain the return distributions:

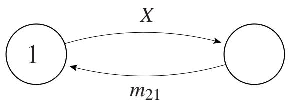

Fig. 6. The Cox-Miller Normal Form for the selected state 1 in the example stochastic transition system

$$m_{01} = \frac{pL_A(z)}{1-(1-p)L_C(z)L_D(z)}, \qquad m_{21} = \frac{pL_A(z)L_D(z)}{1-(1-p)L_C(z)L_D(z)} \tag{4}$$

As was mentioned earlier, in the original system state 1 had only a single out-edge to state 2. This can now be reduced to Cox-Miller Normal Form with the original out-distribution, X, leaving state 1 and the distribution m_{21} comprising the aggregate return distribution (figure 6).

If there had been an out-edge from state 1 to state 0 as well then we would have used m_{01} explicitly to create a 2 out-edge normal form around state 1.

3.3 Solving the General Case

For a general system, we can restate the method above as follows:

Step 1: Create an $n \times n$ transition matrix T where t_{ij} represents the Laplace distributions of each transition.

Step 2: We seek to create a matrix M from T which represents the return transitions (or passage-time transitions) between states using equation (5).

For a given state i, we can construct the following recursive relation involving the return distributions and the modified transition matrix $T^{(i)}$. The transition matrix is modified to ensure termination of the recursive equation, by making the chosen state an absorbing state (by the definition of return distributions):

$$[m_{ji}] = T^{(i)}[m_{ji}{}^{(i)}] \tag{5}$$

where $T^{(i)}$ is constructed from T by replacing row i of T by the vector e_i; and $[m_{ji}{}^{(i)}]$ is constructed from $[m_{ji}]$ by setting $m_{ii}{}^{(i)} = 1$ and $m_{ji}{}^{(i)} = m_{ji}$ for $i \neq j$.

Step 4: The resulting $n-1$ equations are for $n-1$ unknowns and can be solved by standard methods, like Gaussian elimination.

Step 5: Having obtained M (or any subset of columns of M as required), we are now in a position to construct the normal forms for any of the states by using the elements of M as the return distributions.

The above method is $O(n^3)$ for a single state and therefore $O(n^4)$ for the whole system if using a method such as Gaussian elimination to solve the $n-1$ linear equations. However, this can be improved to $O(n^{2.8})$ and $O(n^{3.8})$ respectively for more sophisticated algorithms.

Solving the normal forms for the steady-state distribution does not add to the complexity so we can compare this directly with a Markovian system which would take $O(n^3)$ time to obtain the steady-state distribution for the entire system, if using Gaussian elimination. In general, this makes traditional Markovian techniques an order of n faster than this method, but with the advantage in the stochastic transition system case that general distributions can be manipulated. Also, a matrix of return distributions is found which are potentially much more descriptive than the steady-state distribution, as they can give extra variance and reliability information.

Tantalisingly, it looks likely that the actual number of computations to obtain a steady-state probability for a single state using the matrix method could be less than the number required to obtain an entire distribution from a Markovian system (simply because the size of the matrices differs by a constant). If this is confirmed then there is the possibility of producing partial steady-state distributions for a Markovian system for less computational cost than through solving the entire system.

4 Conclusion

In this paper, we have introduced stochastic transition systems, an algebraic representation of semi-Markov processes. We have constructed mappings from Markovian process algebras, PEPA and MTIPP, as well as a fully generally-distributed stochastic process algebra to their equivalent stochastic transition system.

We have used a process called stochastic aggregation to reduce the stochastic transition systems to a normal form about individual states. These can be easily analysed to obtain the steady-state probability for that state.

Finally, we present a matrix method for generating these normal forms, which has the benefit of being computationally much quicker than a flow graph-based reduction method would be. We obtain time complexities of $O(n^3)$ for single state steady-state probabilities. Therefore, this gives us for the first time the possibility that partial steady-state distributions can be created as required.

We also note that it may be possible to generate individual steady-state probabilities in less time than for solving an entire Markovian steady-state distribution. However, this will need to be comparatively tested on actual tools to confirm the result.

Acknowledgements

Many thanks to Ian Holyer, Rob Pooley, Helen Wilson and anonymous referees for most useful comments. Thanks also go to the Department of Chemical Engineering, University of Colorado at Boulder for providing facility-support while writing this paper.

References

[1] BRADLEY, J. T. *Towards Reliable Modelling with Stochastic Process Algebras*. PhD thesis, Department of Computer Science, University of Bristol, Bristol BS8 1UB, UK, October 1999.

[2] BRADLEY, J. T., AND DAVIES, N. J. Reliable performance modelling with approximate synchronisations. In *Process Algebra and Performance Modelling Workshop* (Zaragoza, September 1999), J. Hillston and M. Silva, Eds., Centro Politécnico Superior de la Universidad de Zaragoza, Prensas Universitarias de Zaragoza, pp. 99–118.

[3] HARRISON, P. G. Laplace transform inversion and passage-time distributions in Markov processes. *Journal of Applied Probability 27* (1990), 74–87.

[4] HERMANNS, H., AND RETTELBACH, M. Syntax, semantics, equivalences, and axioms for MTIPP. In *Process Algebra and Performance Modelling Workshop* (Regensberg, July 1994), U. Herzog and M. Rettelbach, Eds., Arbeitsberichte des IMMD, Universtät Erlangen-Nürnberg, pp. 69–88.

[5] HILLSTON, J. *A Compositional Approach to Performance Modelling*. PhD thesis, Department of Computer Science, University of Edinburgh, Edinburgh EH9 3JZ, UK, 1994. CST–107–94.

[6] HOWARD, R. A. *Dynamic Probabilistic Systems: Markov Models*, vol. 1 of *Series in Decision and Control*. John Wiley & Sons, 1971.

[7] HOWARD, R. A. *Dynamic Probabilistic Systems: Semi-Markov and Decision Processes*, vol. 2 of *Series in Decision and Control*. John Wiley & Sons, 1971.

[8] PLOTKIN, G. D. A structured approach to operational semantics. Technical Report DAIMI FM–19, Department of Computer Science, Aarhus University, DK–8000 Aarhus C, Denmark, 1981.

[9] ROSS, S. M. *Stochastic Processes*. Wiley Series in Probability and Mathematical Statistics. John Wiley & Sons, 1983.

[10] WILSON, H. J., AND BRADLEY, J. T. A note on the proof of reduction of biconnected digraphs to normal forms. CSTR Technical Report CSTR–99–009, Department of Computer Science, University of Bristol, Bristol BS8 1UB, UK, September 1999.

Jeremy Bradley is a member of the Department of Computer Science, University of Durham, South Road, Durham DH1 3LE, UK. E-mail: Jeremy.Bradley@durham.ac.uk

Neil Davies is a member of the Department of Computer Science, University of Bristol, Woodland Road, Bristol BS8 1UB, UK. Email: Neil.Davies@bristol.ac.uk