TU/e EINDHOVEN UNIVERSITY OF TECHNOLOGY

On the random structure of behavioural transition systems

Citation for published version (APA):
Groote, J. F., van der Hofstad, R. W., & Raffelsieper, M. (2014). *On the random structure of behavioural transition systems*. (Computer science reports; Vol. 1401). Technische Universiteit Eindhoven.

Document status and date:
Published: 01/01/2014

Document Version:
Publisher's PDF, also known as Version of Record (includes final page, issue and volume numbers)

Please check the document version of this publication:

• A submitted manuscript is the version of the article upon submission and before peer-review. There can be important differences between the submitted version and the official published version of record. People interested in the research are advised to contact the author for the final version of the publication, or visit the DOI to the publisher's website.
• The final author version and the galley proof are versions of the publication after peer review.
• The final published version features the final layout of the paper including the volume, issue and page numbers.

Link to publication

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

• Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
• You may not further distribute the material or use it for any profit-making activity or commercial gain
• You may freely distribute the URL identifying the publication in the public portal.

If the publication is distributed under the terms of Article 25fa of the Dutch Copyright Act, indicated by the "Taverne" license above, please follow below link for the End User Agreement:

www.tue.nl/taverne

Take down policy
If you believe that this document breaches copyright please contact us at:

openaccess@tue.nl

providing details and we will investigate your claim.

Download date: 28. Feb. 2021

Technische Universiteit Eindhoven
Department of Mathematics and Computer Science

On the Random Structure of Behavioural Transition Systems

Jan Friso Groote, Remco van der Hofstad, Matthias Raffelsieper

14/01

ISSN 0926-4515

All rights reserved
editors: prof.dr. P.M.E. De Bra
prof.dr.ir. J.J. van Wijk

Reports are available at:
http://library.tue.nl/catalog/TUEPublication.csp?Language=dut&Type=ComputerScienceReports&Sort=Author&level=1 and
http://library.tue.nl/catalog/TUEPublication.csp?Language=dut&Type=ComputerScienceReports&Sort=Year&Level=1

Computer Science Reports 14-01
Eindhoven, February 2014

On the Random Structure of Behavioural Transition Systems

Jan Friso Groote⋆, Remco van der Hofstad⋆, Matthias Raffelsieper♯

⋆ Department of Mathematics and Computer Science, Eindhoven University of Technology
P.O. Box 513, 5600 MB Eindhoven, The Netherlands

♯ D-MTEC, ETH Zürich, Weinbergstrasse 56/58, 8092 Zürich, Switzerland

J.F.Groote@tue.nl, R.W.v.d.Hofstad@tue.nl, raffelsm@ethz.ch

Abstract

Random graphs have the property that they are very predictable. Even by exploring a small part reliable observations are possible regarding their structure and size. An unfortunate observation is that standard models for random graphs, such as the Erdös-Rényi model, do not reflect the structure of the graphs that we find in behavioural modelling. In this paper we propose an alternative model, which we show to be a better reflection of 'real' state spaces. We show how we can use this structure to predict the size of state spaces, and we show that in this model software bugs are much easier to find than in the more standard random graph models. Not only gives this theoretical evidence that testing might be more effective than thought by some, but it also gives means to quantify the amount of residual errors based on a limited number of test runs.

1 Introduction

Modelling the behaviour of systems is gaining popularity. An unpleasant side effect is that the transition systems of models of realistic systems easily become forbiddingly large. We ran into such an example while modelling an UART (universal asynchronous receiver/transmitter) for a company called NXP. Using highway search [4], a parallel simulation technique far more efficient than random simulation in finding problematic situations, we did *not* find a suspected error. The question that we needed to answer was how large the probability was that the error really did not occur. A typical derived question that immediately jumps to mind is to estimate the size of the state space.

In order to answer such questions, one can resort to random graphs [2]. The Erdös-Rényi model is a commonly used model. It has a set S of N states (nodes, vertices) and a set of transitions $\rightarrow$. There are two highly similar variants, one where each conceivable edge is present with some probability p, and one where M transitions are chosen out of the N^2 possibilities. It is easy to see that the expected number of

transitions is $pN^2/2$ in the former case where it is exactly M in the latter. Thus $M = pN^2/2$ should yield similar results.

Erdös-Rényi random graphs are a little counterintuitive if it comes to modelling behavioural transition systems. Behavioural transition systems have an initial state and this initial state has outgoing transitions to states that in general also have outgoing transitions. In the Erdös-Rényi random graph the initial state may not have outgoing transitions (actually with a fairly high probability $e^{-\lambda}$ where λ is the fan-out, i.e., the expected number of transitions leaving a state). Therefore, we choose a slightly different model, where each state has a fixed number λ of outgoing transitions each of which goes to randomly selected states of the transition system. All choices are made independently of each other.

Given this model of a random transition system we estimate the size of a transition system by a random walk through the graph. By random simulation we provide evidence that these estimates are very good. However, by applying this technique to realistic models (e.g., Firewire P1394 protocol [7]) it becomes obvious that the structure of these random graphs is not really a reflection of a 'real state space'.

As an alternative model for the structure of realistic systems, we propose to use the Cartesian product of P parallel random transition systems, reflecting that a realistic system often consists of P more or less independent components. One could not only think of the components as independent parallel processes, but one can also consider the behaviour of subtasks or even variables as potential parallel components.

We develop techniques to estimate the sizes and fanout of the different components. Again, using random simulation we verify that these estimation techniques are correct and precise. More importantly, we estimate the sizes of 'realistic state spaces' and find that these are far better than those we obtain using the 'single threaded' random model. There are also some disadvantages. In particular, the predictions are less stable and the calculational effort for the estimation is higher.

Our experiments provide evidence that P-parallel random transition systems are a good representation of 'realistic' state spaces. Of course, more experiments are required to corroborate this, and these experiments may lead to further refinements of the model. But having such a 'realistic' random model is really a great asset, because it enables the use of the power of random analysis to substantially increase our insight in the behaviour of real systems.

As an illustration of the potential power of the P-parallel model we apply it to the question how effective testing is. In our experience it is remarkably easy to detect a known error by running a random test. According to the single threaded random model this is not possible. The probability of hitting an erroneous state by a random walk is far too small. However, if the error occurs in one of the states of one of the P parallel components, it is far easier to find it. Even stronger, if we know the sizes of the different components, we can come up with fairly small numbers of required test runs to guarantee with a given confidence that realistic systems are error free.

We still have not answered the question of the effectiveness of highway search. But we believe that the insight that P-parallel systems reflect real systems quite well, together with the non-trivial mathematics required to employ this insight deserves to be known by a wider audience.

Related Work.

As far as we know, there is not much work on the random structure of transition systems representing behaviour. The following is what we are aware of. Estimating the size of a Petri Net's state space has been investigated in [9]. That work makes explicit use of the structure of a Petri Net and is only applicable when the Petri Net is constructed from a set of supported building blocks.

A more general approach was presented in [12] where, as in our work, a state space is seen as a directed graph. However, the authors do not compute an estimate, but instead only classify state spaces into one of three classes: small models, large models, and models that are too large and hence out of reach. To do this, they propose two techniques: The first is based on a capture-recapture sampling similar to what is presented in this paper in Section 3. However, the analysis does not take the process of sampling into account and simply assumes that two independent samples can be obtained. From these two samples, a computation similar to the Lincoln Index [11], which was originally developed to estimate animal populations, is performed and based on this value a class is assigned to the model. The second technique uses a number of different observations made during a breadth-first exploration of the state space as input to existing classification methods such as classification trees and neural networks.

Inspired by this work, the authors of [3] present a method to compute the estimated state space size. There, the observed measure is the size of the breadth-first frontier that is still to be explored in relation to the number of states that have already been explored. By visual inspection, the authors determine that this curve should be approximated with a quadratic function and use least-squares fitting to compute the parameters and thereby an estimate for the state-space size.

2 Random State Spaces

In this section we define the basic notions that we employ. We use directed graphs or transitions systems without labels, as we do no need the labels in our exposition.

A *state space* is seen as a graph $G = (S, \to)$, with S being an arbitrary set of *states* (nodes, vertices) and $\to \subseteq S \times S$ being a multi-set of *transitions* (edges). If $(s, s') \in \to$ holds, we generally denote this by $s \to s'$. For the edges in the set $\to$ we assume that every state has a fixed degree of outgoing edges, i.e., there exists a fixed $\lambda \in \mathbb{N}$ such that $|\{s' \mid s \to s'\}| = \lambda$ for all $s \in S$ (where $\{s' \mid s \to s'\}$ is a multi-set) and $|E|$ denotes the size of a set E. If $s \to s'$, then s is called the *source* and s' the *target* state of that edge. In a random state space it is assumed that for every such edge, given its source state s, every other state s' is equally likely to be the target state. Furthermore, we define $N = |S|$ and $M = |\to|$ to denote the number of states and transitions, respectively.

A tuple $T = (G, s_0)$ is called a *random transition system*, where $G = (S, \to)$ is a random state space as described above and $s_0 \in S$ is an arbitrary, randomly chosen *initial state*. For such a random transition system, only the part reachable from the initial state is of interest, i.e., those states $s' \in S$ for which $s_0 \to^* s'$ holds (where $\to^*$ denotes the reflexive transitive closure of $\to$, inductively defined by $s \to^* s$ and

if $s \to s'$ and $s' \to^* s''$, then $s \to^* s''$). Note that the number of reachable states is generally smaller than N.

This paper considers state spaces being the graph product of two or more random transition systems. Since taking the graph product is associative, we only consider the case of two random transition systems, which can then be repeated for more components. Thus, a *product transition system* $T_{1\times 2} = (G, s_0)$ with graph $G = (S, \to)$ and initial state $s_0 \in S$ is assumed to be composed from two random transition systems $T_1 = (G_1, s_{1,0})$ and $T_2 = (G_2, s_{2,0})$, with $G_1 = (S_1, \to_1)$, $G_2 = (S_2, \to_2)$, such that $S = S_1 \times S_2$, $s_0 = (s_{1,0}, s_{2,0})$, and $(s_1, s_2) \to (s_1', s_2')$ iff either $s_1 \to_1 s_1'$ and $s_2 = s_2'$, or $s_1 = s_1'$ and $s_2 \to_2 s_2'$.

Note that it is assumed that the states in the product transition system $T_{1\times 2}$ are opaque, i.e., from a state $s = (s_1, s_2) \in S$ the individual components s_1 and s_2 of the state cannot be recovered. Note also that we do not consider 'synchronisation', i.e., a transition in the product transition system which consists of the simultaneous occurrence of transitions in the constituent transition systems. Our random approximations of realistic state spaces do not contain such synchronised transitions.

3 Estimation Based On Duplicates

In this section, a technique is described to estimate the size of a single random transition system $T = (G, s_0)$ with graph $G = (S, \to)$. For this purpose, the process of exploring the state space is analysed, where starting from the initial state s_0 iteratively edges are explored that have their source state in the already explored part of the state space. Thus, a target state of an edge that is being explored can either be a state that has already been explored previously, or it is a state that is new, i.e., has not been seen previously. Note that the method in this section does not rely on a particular state space exploration strategy (e.g., breadth-first or depth-first).

3.1 A stochastic model

We develop a stochastic model where we introduce random variables Y_k, which represent the number of unique states seen after exploring k transitions. Then, the probability distribution of the state space size after exploring m transitions is

$$\mathbb{P}[N = n \mid \bigwedge_{k=0}^{m} Y_k = i_k],$$

where i_k is the observed number of unique states seen after exploring k transitions. Applying Bayes' law, this probability can be rewritten as follows:

$$\begin{aligned}\mathbb{P}[N = n \mid \bigwedge_{k=0}^{m} Y_k = i_k] &= \frac{\mathbb{P}[N = n \wedge \bigwedge_{k=0}^{m} Y_k = i_k]}{\mathbb{P}[\bigwedge_{k=0}^{m} Y_k = i_k]} \\ &= \frac{\mathbb{P}[\bigwedge_{k=0}^{m} Y_k = i_k \mid N = n]\mathbb{P}[N = n]}{\sum_{n=i_m}^{\infty} \mathbb{P}[\bigwedge_{k=0}^{m} Y_k = i_k \mid N = n]\mathbb{P}[N = n]} \qquad (1)\end{aligned}$$

In the above equation (1), the probability $\mathbb{P}[N = n]$ is the so-called *a-priori* probability for the size of the state space. To characterise the probabilities $\mathbb{P}[\bigwedge_{k=0}^{m} Y_k = i_k \mid N = n]$, we use the following recursive identity:

$$\mathbb{P}[\bigwedge_{k=0}^{m} Y_k = i_k \mid N = n] = \\ \mathbb{P}[Y_m = i_m \mid \bigwedge_{k=0}^{m-1} Y_k = i_k \wedge N = n]\, \mathbb{P}[\bigwedge_{k=0}^{m-1} Y_k = i_k \mid N = n] \quad (2)$$

Thus, we need to analyse the first factor at the right-hand side of equation (2) further. In case $m = 0$, then we find that

$$\mathbb{P}[\bigwedge_{k=0}^{0} Y_k = i_k \mid N = n] = \begin{cases} 1 & \text{if } i_0 = 1 \\ 0 & \text{otherwise,} \end{cases}$$

since only the initial state is seen in the beginning.

For $m > 0$, we make a case distinction based on whether the target state of the newly explored edge has been seen before or not. In case $i_m = i_{m-1}$, i.e., the target state has already been seen before, then one of the i_{m-1} already seen states has been chosen:

$$\mathbb{P}[Y_m = i_m \mid \bigwedge_{k=0}^{m-1} Y_k = i_k \wedge N = n] = \frac{i_{m-1}}{n} = \frac{i_m}{n} \quad (3)$$

Otherwise, if the target state of the newly explored edge has not been seen before, then $i_m = i_{m-1} + 1$ holds and we have chosen one of the $n - i_{m-1}$ states we have not yet seen, giving the following equality:

$$\mathbb{P}[Y_m = i_m \mid \bigwedge_{k=0}^{m-1} Y_k = i_k \wedge N = n] = \frac{n - i_{m-1}}{n} = \frac{n - i_m + 1}{n} \quad (4)$$

Next, we solve the recurrence in equation (2) using equations (3) and (4). To do so, we introduce variables $q_j^{(m)}$ for $1 \leq j \leq i_m$ to represent the multiplicity of number j in the sequence $i_0, i_1, \ldots, i_m$ of length $m+1$. For example, in the sequence $1, 1, 2, 3, 3, 4$ it holds that $q_1^{(5)} = 2$, $q_2^{(5)} = 1$, $q_3^{(5)} = 2$, and $q_4^{(5)} = 1$. It should be noted that $\sum_{j=1}^{i_m} q_j^{(m)} = m + 1$ always holds. Using these variables, the right hand side of equation (2) becomes

$$\frac{\prod_{j=1}^{i_m}(n - j + 1) \prod_{j=1}^{i_m} j^{q_j^{(m)} - 1}}{n^{m+1}}. \quad (5)$$

Substituting equation (5) into (1) gives the following result, where remarkably enough the variables $q_j^{(m)}$ disappear. Apparently, only the information about the number of unique states (the i_k) and the total number of states explored (which is $m + 1$, i.e., the number of target states of explored edges plus the initial state) is required:

$$\mathbb{P}[N = n \mid \bigwedge_{k=0}^{m} Y_k = i_k] = \frac{\left(\prod_{j=1}^{i_m}(n - j + 1)\right) \mathbb{P}[N = n]/n^{m+1}}{\sum_{r=i_m}^{\infty} \left(\prod_{j=1}^{i_m}(r - j + 1)\right) \mathbb{P}[N = r]/r^{m+1}} \quad (6)$$

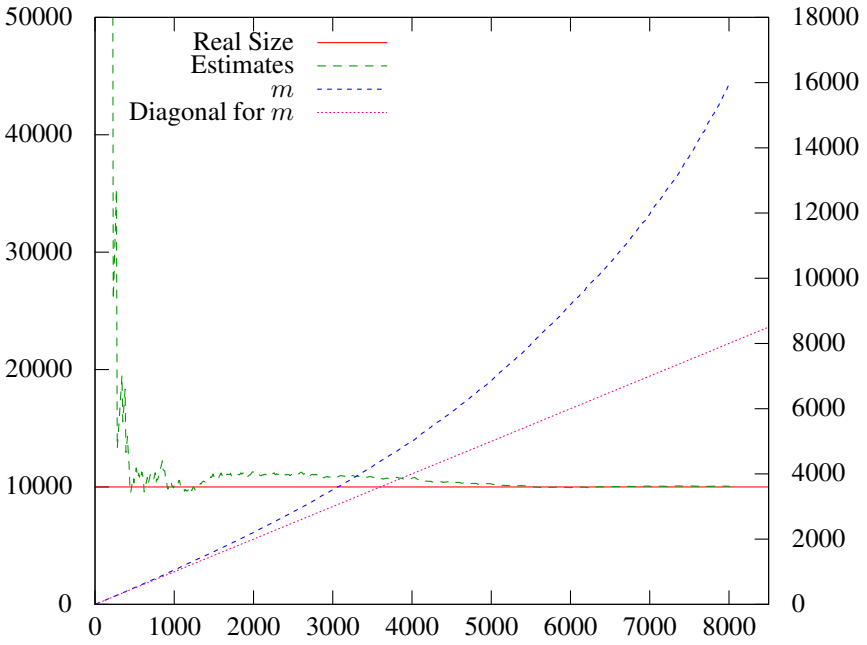

Figure 1: Estimates using equation (8) for an example random transition system with $10,000$ states and a fan-out of 2. On the x-axis the number i_m of unique states observed are depicted. On the y-axis, the estimated size is shown at the left, and the number of explored transitions m and the diagonal are shown at the right.

Often, an upper bound $\overline{n}$ on the total number of states can be obtained. Furthermore, the maximally observed number of unique states i_m is a lower bound for the total size N. Hence, we assume that N is a priori uniformly distributed in the interval $[i_m, \overline{n}]$. In this case, equation (6) reduces to

$$\mathbb{P}[N = n \mid \bigwedge_{k=0}^{m} Y_k = i_k] = \frac{\left(\prod_{j=1}^{i_m}(n-j+1)\right)/n^{m+1}}{\sum_{r=i_m}^{\overline{n}}\left(\prod_{j=1}^{i_m}(r-j+1)\right)/r^{m+1}}, \tag{7}$$

which gives the following expected size of the state space:

$$\mathbb{E}[N \mid \bigwedge_{k=0}^{m} Y_k = i_k] = \frac{\sum_{n=i_m}^{\overline{n}} n\left(\prod_{j=1}^{i_m}(n-j+1)\right)/n^{m+1}}{\sum_{r=i_m}^{\overline{n}}\left(\prod_{j=1}^{i_m}(r-j+1)\right)/r^{m+1}} \tag{8}$$

3.2 Simulation experiments

We want to establish the effectiveness of the estimation procedure by predicting the sizes of a randomly generated state spaces and comparing the results with the actual sizes used to generate them.

For the experiments with the estimation procedure presented in the previous section, we explored the example graphs in a breadth-first fashion, as this is the commonly used strategy of many model checking tools, an example being the tool LPS2LTS contained in the mCRL2 toolset [5]. When used in verbose mode, the tool will output the currently explored number of states and transitions, which therefore allows to apply equation (8) to estimate the total size of the state space.

We present here the results on a random transition system with $10,000$ states, each with 2 outgoing transitions. After randomly designating an initial state, $8,011$ states were reachable. To compute the estimates, we used $\overline{n} = 1,000,000$ as upper bound on the size of the state space.

The estimation results are shown in figure 1. The x-axis shows the number of unique states seen at a certain point, i.e., the maximal i_m used in that computation. On the y-axis, the result of evaluating equation (8) is depicted. The values of the maximal i_m were increased in steps of 10 and two adjacent points were connected by a straight line segment. It can be observed that the estimation yields results close to the actual value of $10,000$ after having observed a few hundred unique states. After that, it slightly overshoots, but always stays below $11,500$ estimated states. From this we conclude that our estimates are quite accurate.

3.3 Application to the firewire protocol

We use a description of a real time bus access in the firewire or P1394 protocol provided in [7] to observe what happens when we predict the number of states for a realistic protocol assuming that it is randomly generated. The protocol consists of two protocol entities that alternatingly obtain access to a data bus resolving contention conflicts on the way.

The typical values for i_m and m are given in table 1. The estimate for the expected number of states N and the number of reachable states $N_{\mathrm{reachable}}$ are also provided. The actual number of reachable states is 188569 and there are 340608 transitions. This leads to an average fan-out of 1.8.

In table 1 we observe that the estimates for $N_{\mathrm{reachable}}$ while traversing the state space structurally underestimate the actual number of states and they only approache the actual number of states when all states have been traversed. This is quite different from what we observe in figure 1. But this pattern is very similar to what we see in other state spaces representing real systems. From this we conclude that our random model does not represent real systems sufficiently well.

3.4 Application to a product graph

The previous example clearly shows that we need an alternative random model to resemble the structure of real systems. Therefore, we looked into Cartesian products of random transition systems. If we estimate the size of a product graph constructed from two random transition systems as if it were a single random transition system then we obtain very similar results as for the model of the firewire protocol.

We report here one experiment, typical for all those that we carried out. It consists of the product of two random transition systems. The first transition system has

i_m	11489	21717	51704	102265	150728	188569
m	13889	26669	79388	175029	273051	340608
N	35435	62637	85455	146869	204586	256631
$N_{\text{reachable}}$	25954	45878	62591	107572	149846	187965

Table 1: Estimates for the number of states of the firewire protocol. The actual number of reachable states is 188569. The estimates for N under the assumption that the state space has a random structure are not very accurate.

100 states and a fan-out of 2, whereas the second transition system has 200 states and a fan-out of 3. In the product of the two, $14,586$ of the total $20,000$ states are reachable. The results when viewing the product of the two random transition systems as a single random transition system are shown in figure 2, where again the upper bound on the state space size was chosen as $1,000,000$.

In the figure, it can be seen that the estimates are always below the real value (except for the first estimate, see the dashed line close to the y-axis), and increase rather linearly. Also, the estimates are only slightly above the number of unique states already seen, so the estimate expects that almost all of the state space has been considered.

The reason why this happens is that in the product graph states are seen more than once (i.e., the difference between the number of explored transitions m and the number of states seen i_m) due to the commuting transitions of the two transition systems, i.e., it does not matter when first doing a step in the first component and then in the second or vice versa, the reached state is always the same. The estimation procedure is sent astray by this behaviour, as it expects duplicates to occur this often only at the end, when almost all of the state space has been explored. This is also demonstrated by the curve for the number of explored transitions m, shown in figure 2, which outgrows the number of seen states (i.e., the diagonal for m in that figure) very early.

The important observation is that the predictions for the size of the firewire protocol and the product graph are very similar. This leads to the assumption that the transition graphs of real systems are far better represented by products of random transition systems. In the next section, we work out a way to estimate the size of product graphs.

4 Estimation of the Sizes of Product Transition Systems

Here we assume that we are dealing with a transition system that is the product of two transition systems. Since we assume that in a product transition system the identity of a component state cannot be recovered, there is no way to obtain the original graphs. If we were able to recover the constituent graphs, we could have applied the technique of section 3 to the components and easily derive a prediction for the size of the whole transition system.

Our estimation technique is completely different from that of section 3. We require that the exploration of a state space is performed in a breadth-first fashion, i.e., first all states at a certain distance from the initial state are considered, before dealing with

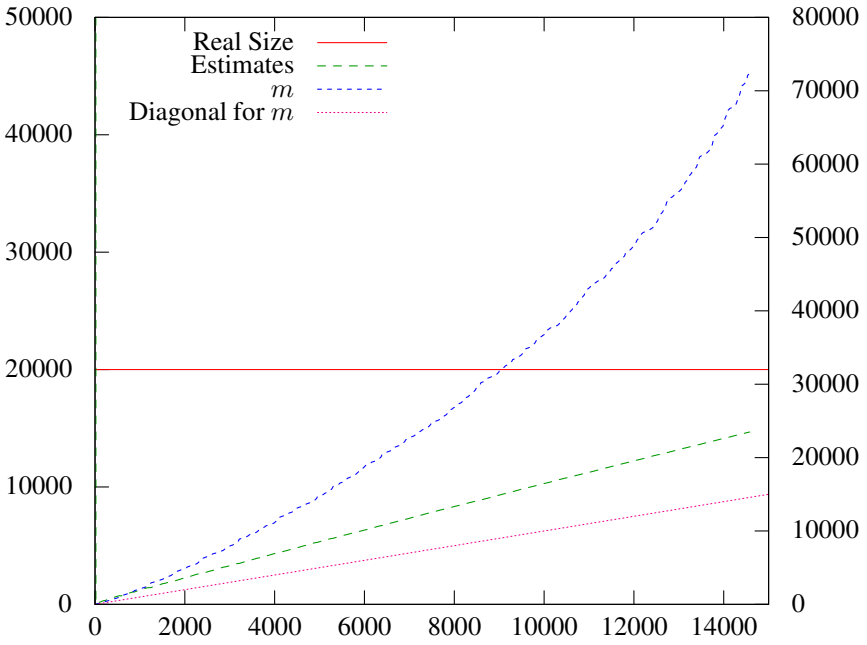

Figure 2: Estimates using equation (8) for a product transition system of 100×200 states with fan-outs of 2 and 3, respectively. On the y-axis, the estimated size is shown at the left. The number of explored transitions m and a diagonal are shown at the right.

those at higher distances. This allows to consider *layers* of a transition system $T = (G, s_0)$ with $G = (S, \to)$. We define the layer $\partial B_T(j)$ at some distance $j \in \mathbb{N}$ by

$$\partial B_T(j) = \{s \in S \mid s_0 \to^j s \wedge \forall k < j : s \notin \partial B_T(k)\}$$

where $s_0 \to^j s$ means that state s is reachable from state s_0 in exactly j steps. We also call $\partial B_T(j)$ the boundary ball on intrinsic distance j. Note that by the above definition, every state s reachable from the initial state s_0 is contained in exactly one layer, namely that with the minimal distance j.

The set of those states that have been seen up to some distance $j \in \mathbb{N}$ is defined as $B_T(j) = \bigcup_{i=0}^{j} \partial B_T(i)$ and is called the *ball* of radius j. To obtain the layer $\partial B_T(j)$, only states from the layer $\partial B_T(j-1)$ have to be considered, since otherwise, if there was an edge from a state $s \in \partial B_T(j-k)$ with $k > 1$ to a state $s' \in \partial B_T(j)$, then this would imply $s' \in \partial B_T(j-k+1)$, which would contradict $s' \in \partial B_T(j)$.

4.1 Estimating the size of a single component

We first concern ourselves with the estimation of the size of a single random transition system based on balls and layers. In section 4.2 we use this result to estimate the sizes of the transition systems in parallel products of transition systems. Breadth-first

generation of a state space explores edges having their source state in the current layer and adds the target states to the current ball. Thus, to obtain a layer at distance $j+1$, a total of $\lambda|\partial B_T(j)|$ edges are explored. This observation is used in the following lemma, which gives a recursive formula for the number of edge-explorations performed to construct the ball with radius $j+1$.

Lemma 4.1. To obtain $B_T(j+1)$, a total of $\lambda|B_T(j)|$ edges are explored.

Proof. We perform induction on j. If $j=0$, then only the initial state is contained, i.e., $B_T(j)=\{s_0\}$. Thus, λ successors of the initial state are explored in the first layer, giving a total of $\lambda|B_T(0)|=\lambda\cdot 1=\lambda$ edges explored to obtain $B_T(1)$. Otherwise, if $j>0$, then we use the observation that $B_T(j+1)=B_T(j)\cup\partial B_T(j+1)$. Hence, first a number of edges is explored to obtain $B_T(j)$, which by the induction hypothesis is $\lambda|B_T(j-1)|$. To obtain the layer $\partial B_T(j+1)$, it was already observed above that $\lambda|\partial B_T(j)|$ edges are explored. Thus, the total number of edges explored is $\lambda|B_T(j-1)|+\lambda|\partial B_T(j)|=\lambda|B_T(j)|$. □

We are interested in the expected size of the reachable state space, which is the same as the size of the ball with maximal radius. Thus, the expected size of the balls should be investigated.

Definition 4.2. We define $G(j)=\mathbb{E}[|B_T(j)|]$ to be the expected size of the breadth-first graph with states at a distance $\leq j$ from the initial state and $R(j)=\mathbb{E}[|\partial B_T(j)|]=G(j)-G(j-1)$, where $G(-1)=0$.

We use the convention that the subscript of B_T carries over to G and R. For example, we write $G_{1\times 2}(j)$ for $\mathbb{E}[|B_{T_{1\times 2}}(j)|]$

To analyse the function $G(j)$, we introduce random variables X_i that denote the size of the partial state space after exploring i edges. Let n_i for $i=1,2,\ldots$ denote the number of unique states observed after exploring i edges. Thus, $n_0=1$ and for n_{i+1} we have either $n_{i+1}=n_i$ in case the target state of the additionally explored edge was already in the explored part of the state space, or $n_{i+1}=n_i+1$ if the target state of the edge is new. Since the target state of an edge is picked uniformly at random, the probability of the first case ($n_{i+1}=n_i$) to occur is

$$\mathbb{P}(X_{i+1}=n_{i+1}\mid \bigwedge_{l=0}^{i} X_l=n_l)=\frac{n_i}{N}=\frac{n_{i+1}}{N},$$

which amounts to the probability to pick one of the $n_i=n_{i+1}$ states that were already explored from the total N states. In the second case, where $n_{i+1}=n_i+1$, the probability is

$$\mathbb{P}(X_{i+1}=n_{i+1}\mid \bigwedge_{l=0}^{i} X_l=n_l)=1-\frac{n_i}{N}=\frac{N-n_{i+1}+1}{N},$$

where a state is picked that is not among the $n_i=n_{i+1}-1$ already explored states.

The expected value of these random variables can be computed as

$$\begin{aligned}
\mathbb{E}[X_{i+1}] &= \sum_{k=0}^{\infty} k\mathbb{P}(X_{i+1} = k) \\
&= \sum_{k=0}^{\infty} k\frac{k}{N}\mathbb{P}(X_i = k) + \sum_{k=0}^{\infty} k\frac{N-k+1}{N}\mathbb{P}(X_i = k-1) \qquad (9) \\
&= \sum_{k=0}^{\infty} k\frac{k}{N}\mathbb{P}(X_i = k) + \sum_{k=0}^{\infty} (k+1)\frac{N-k}{N}\mathbb{P}(X_i = k) \\
&= \sum_{k=0}^{\infty} k(1-\frac{1}{N})\mathbb{P}(X_i = k) + \sum_{k=0}^{\infty} \mathbb{P}(X_i = k) \\
&= \sum_{k=0}^{\infty} k(1-\frac{1}{N})\mathbb{P}(X_i = k) + 1 \\
&= \frac{N-1}{N}\mathbb{E}[X_i] + 1. \qquad (10)
\end{aligned}$$

In equation (9) we used the above observations that either an already explored state is reached or a new state was explored. Solving the recurrence equation (10), using the boundary condition that $\mathbb{E}[X_0] = 1$, gives a closed formula:

$$\mathbb{E}[X_i] = N\left(1 - \left(\frac{N-1}{N}\right)^{1+i}\right). \qquad (11)$$

This equation can also be used to predict N in table 1 with almost equal estimates for N.

Another use of equation (11) is to derive the number of reachable states s. All reachable states have been explored if there are no other states that have been visited. Then there are λs visits, as for each explored state each outgoing transition is investigated. So, equation (11) becomes

$$s = N\left(1 - \left(\frac{N-1}{N}\right)^{1+\lambda s}\right). \qquad (12)$$

From this s can easily be solved, although care is required as there is also a small negative solution for s.

Equation (11) and lemma 4.1 can also be used to derive the following recursive formula for the function $G(j)$, i.e., the expected size of the ball with radius j for a random transition system, where N and λ denote the total size and the fan-out of each state, respectively:

$$G(j+1) = \mathbb{E}[X_{\lambda|B_T(j)|}] = N\left(1 - \left(\frac{N-1}{N}\right)^{1+\lambda G(j)}\right). \qquad (13)$$

Equation (13) gives a means to compute the size N given two consecutive balls, given that we can easily estimate λ as the average fan-out in each state. With more balls available, N can be estimated using the least square method.

4.2 Product Graph Size Estimation

As the estimates of the size of the state spaces are way off if we assume that they are homogenous random state spaces, we are interested in viewing a state space as the product of two random state spaces. We assume that the two components cannot be distinguished in the product graph and therefore we need to formulate a relation between the observable ball and layer sizes of the product graph and the component sizes. For this purpose, we note that due to the definition of the graph product, which only performs a transition in one of the components, a state in the layer of depth j can be reached by either only performing j steps in the first component, or $j-1$ steps in the first component and one step in the second component, etc. Furthermore, steps from different components are independent of each other, i.e., if $(s_1, s_2) \rightarrow_2 (s_1, s_2') \rightarrow_1 (s_1', s_2')$, then also $(s_1, s_2) \rightarrow_1 (s_1', s_2) \rightarrow_2 (s_1', s_2')$, where $\rightarrow_i$ is a step done by component i. Hence, we can re-order the steps such that first all steps in the first component are performed, and then all steps in the second component. Thus, in the product graph, the following equation holds:

$$|\partial B_{T_1 \times T_2}(j)| = \sum_{k=0}^{j} |\partial B_{T_1}(k)| \cdot |\partial B_{T_2}(j-k)|. \tag{14}$$

From equation (14) for the expected value $R_{1\times 2}(j)$ of the product graph we derive:

$$R_{1\times 2}(j) = \sum_{k=0}^{j} R_1(k) \cdot R_2(j-k). \tag{15}$$

In the following, the expected sizes of the component layers, R_1 and R_2, are considered in more detail. Let $i \in \{1, 2\}$ and recall that $R_i(j+1) = G_i(j+1) - G_i(j)$. Using equation (13), we can also obtain a recursive formula for the component layer sizes:

$$\begin{aligned}
R_i(j+1) &= G_i(j+1) - G_i(j) \\
&= N_i \left[\left(\frac{N_i - 1}{N_i} \right)^{1+\lambda_i G_i(j-1)} - \left(\frac{N_i - 1}{N_i} \right)^{1+\lambda_i G_i(j)} \right] \\
&= N_i \left(\frac{N_i - 1}{N_i} \right)^{1+\lambda_i G_i(j-1)} \left[1 - \left(\frac{N_i - 1}{N_i} \right)^{\lambda_i (G_i(j) - G_i(j-1))} \right] \\
&= (N_i - G_i(j)) \left[1 - \left(\frac{N_i - 1}{N_i} \right)^{\lambda_i R_i(j)} \right] \\
&= \left(N_i - \sum_{k=0}^{j} R_i(k) \right) \left[1 - \left(\frac{N_i - 1}{N_i} \right)^{\lambda_i R_i(j)} \right]
\end{aligned} \tag{16}$$

Substituting equation (16) into equation (15) for R_1 and R_2 yields an equation for the observable layer sizes of the product graph, in which the values of N_1, N_2, λ_1, and λ_2 are unknown. Note however that the combined fan-out $\lambda_1 + \lambda_2$ can be observed

Depth	0	1	2	3	4	5	6	7	8	9	10
$\lvert\partial\mathbf{B}_{\mathbf{T_1}}(\mathbf{j})\rvert$	1	2	4	7	13	17	18	10	4	1	1
$\lvert\mathbf{B}_{\mathbf{T_1}}(\mathbf{j})\rvert$	1	3	7	14	27	44	62	72	76	77	78
$\lvert\partial\mathbf{B}_{\mathbf{T_2}}(\mathbf{j})\rvert$	1	3	8	22	46	54	41	12			
$\lvert\mathbf{B}_{\mathbf{T_2}}(\mathbf{j})\rvert$	1	4	12	34	80	134	175	187			

Table 2: Layer and ball sizes for two example random graphs, with $N_1 = 100$, $\lambda_1 = 2$, $N_2 = 200$, and $\lambda_2 = 3$.

in the product graph, since this is the fan-out of each state in that graph. Therefore, one of the fan-out values can be eliminated, for example by replacing λ_2 with $\lambda - \lambda_1$, where $\lambda = \lambda_1 + \lambda_2$ is the observed constant fan-out in the product graph. By observing the sizes of three layers, three equations can be obtained from which the values of N_1, N_2 and λ can be solved, which is enough to provide the desired estimates. As before, if more layers are available, the parameters can be estimated using the least square method.

It should be noted that the above can easily be extended to more than two components. For example, in case of a product graph consisting of three random transition systems, equation (14) becomes:

$$|\partial B_{T_1 \times T_2 \times T_3}(j)| = \sum_{k=0}^{j} \sum_{l=0}^{j-k} |\partial B_{T_1}(k)| \cdot |\partial B_{T_2}(l)| \cdot |\partial B_{T_3}(j-k-l)|. \qquad (17)$$

4.3 Simulation experiments

The presented approximation technique, combining equations (15) and (16), was implemented in the Mathematica [10] and evaluated on a number of randomly generated state spaces to see whether the approximation technique would result in decent estimates.

As a first example, the product graph of section 3.2 is considered. In the first component, the total number of states is $N_1 = 100$ and the fan-out of each state is $\lambda_1 = 2$, while in the second component a total of $N_2 = 200$ states exist, each having a fan-out of $\lambda_2 = 3$. So, $\lambda = 5$ and $N = 20,000$. The ball and layer sizes of the two component graphs are shown in table 2 and the ball and layer sizes of their product are shown in table 3. It should be noted that not all states in the component graphs are reachable from the initial state, therefore the maximal ball size is smaller than the total number of states.

In the experiments, the sizes of the layers in the columns labeled $|\partial\mathbf{B}_{\mathbf{T_1}\times\mathbf{T_2}}(\mathbf{j})|$ in table 3 were used to solve three successive layer equations built according to equations (15) and (16) for the unknowns N_1, N_2, N and λ_1. These results are shown in table 4. There, in each of the columns 2 through 9 the solution for the layer distances shown in the first row of that column are shown in rows 2 through 4. Solutions for the state space sizes N_1 and N_2 were limited to be in the range $[1.1, 100, 000]$ and solutions

Depth	$\lvert\partial\mathbf{B}_{\mathbf{T_1}\times\mathbf{T_2}}(\mathbf{j})\rvert$	$\lvert\mathbf{B}_{\mathbf{T_1}\times\mathbf{T_2}}(\mathbf{j})\rvert$	Depth	$\lvert\partial\mathbf{B}_{\mathbf{T_1}\times\mathbf{T_2}}(\mathbf{j})\rvert$	$\lvert\mathbf{B}_{\mathbf{T_1}\times\mathbf{T_2}}(\mathbf{j})\rvert$
0	1	1	9	2,308	6,385
1	5	6	10	2,619	9,004
2	18	24	11	2,384	11,388
3	57	81	12	1,696	13,084
4	156	237	13	910	13,994
5	346	583	14	384	14,378
6	660	1,243	15	143	14,521
7	1,118	2,361	16	53	14,574
8	1,716	4,077	17	12	14,586

Table 3: Layer and ball sizes for the product graph of the two example graphs presented in table 2.

Depths	1–3	2–4	3–5	4–6	5–7	6–8	7–9	8–10	9–11
$\mathbf{N_1}$	*100,000*	44.6	42.3	195.9	99.9	112.5	114.5	123.2	563.92
$\mathbf{N_2}$	*95.8*	268.4	271.1	174.1	192.1	195.3	174.5	158.4	32.0
$\mathbf{N}$	*9,580,000*	11,971	11,468	34,106	19,191	21,971	19,980	19,515	18,045
$\lambda_\mathbf{1}$	*2.42*	1.95	1.94	1.99	2.12	2.04	2.14	2.18	2.38

Table 4: Results for solving for N_1, N_2, N, and λ_1 in the product graph shown in table 3.

for the fan-out λ_1 were limited to the range $[0.0, \lambda/2]$ as we can assume that $\lambda_1 \leq \lambda_2$ by symmetry. In the second column, it can be seen that one of the boundaries was reached, for this reason the column is shown in italics since Mathematica [10] stopped here.

For the remaining cases, the solutions are rather close to the actual values. Especially in the region of depths 5–9 the results are very good, but they degrade after that when depth 10 is reached. A reason for this seems to be that the predicted layer sizes and the actually observed ones start to differ by larger amounts from this distance on, as can be seen in figure 3 which compares these two values. The figure shows that in the example product graph, the layer at distance 10 is larger than predicted, and that the layers thereafter decrease faster in size than the prediction.

In the second experiment, we added another component to the product graph shown in table 3, i.e., we now considered a state space constructed from three random transition systems. This added component has a total of $N_3 = 400$ states and a fan-out of $\lambda_3 = 4$. The layer and ball sizes of this added component are shown in table 5, while the combined layer and ball sizes of the product of these three components are shown in table 6.

In the product graph, it can be observed that the combined fan-out is $\lambda = 9$. Hence, one of the fan-out values can be derived from the others. We opted to set $\lambda_3 = \lambda - \lambda_1 - \lambda_2$, therefore the combination of three instances of equation (16) (one for each component) and a two-fold application of equation (15) contains the five

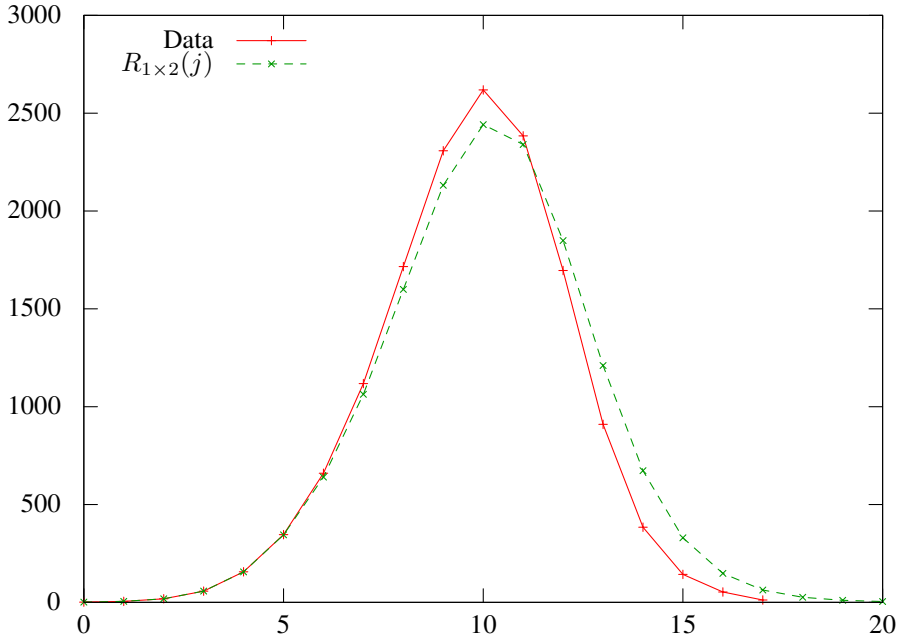

Figure 3: Comparison of observed layer sizes and predicted layer sizes for the product graph of table 3.

Depth	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
$\lvert\partial\mathbf{B}_{\mathbf{T_3}}(\mathbf{j})\rvert$	1	4	15	54	145	148	25	3
$\lvert\mathbf{B}_{\mathbf{T_3}}(\mathbf{j})\rvert$	1	5	20	74	219	367	392	395

Table 5: Layer and ball sizes for an example random graph, with $N_3 = 400$ and $\lambda_3 = 4$.

unknowns N_1, N_2, N_3, λ_1, and λ_2. When solving for these values using the tool Mathematica [10], where the solutions for N_1, N_2, and N_3 were restricted to the range $[1.1, 1 \times 10^7]$ and the solutions for λ_1 and λ_2 were restricted to the ranges $[0, \lambda/3]$ and $[0, \lambda/2]$, respectively, we obtained the results shown in table 7. For the combination of depths 10–14 Mathematica reached the lower boundary of the range allowed for N_3. Since it stops there, the results for this particular combination are shown in italics. In case of the combinations between depth 5 and depth 13 one can observe that the results are rather close to the real values. It should be remarked that for the combination of depths 8–12, the values are swapped, i.e., the solution found has assigned the different sizes differently to the individual components. The solutions obtained for the remaining combinations are further away from the real values.

A comparison between the layer sizes measured in this product of two random state spaces and the predicted layer sizes when setting the parameters N_1, N_2, N_3, λ_1, λ_2, and λ_3 to their actual values is shown in figure 4. Also in this case, the actual layer

Depth	$\lvert\partial\mathbf{B_T(j)}\rvert$	$\lvert\mathbf{B_T(j)}\rvert$
0	1	1
1	9	10
2	53	63
3	258	321
4	1,069	1,390
5	3,670	5,060
6	10,837	15,897
7	28,429	44,326
8	66,293	110,619
9	136,319	246,938
10	248,942	495,880
11	409,052	904,932
12	606,971	1,511,903

Depth	$\lvert\partial\mathbf{B_T(j)}\rvert$	$\lvert\mathbf{B_T(j)}\rvert$
13	803,438	2,315,341
14	925,793	3,241,134
15	903,053	4,144,187
16	726,676	4,870,863
17	473,520	5,344,383
18	248,477	5,592,860
19	108,447	5,701,307
20	41,827	5,743,134
21	14,311	5,757,445
22	3,530	5,760,975
23	459	5,761,434
24	36	5,761,470

Table 6: Layer and ball sizes for the product graph $T = T_1 \times T_2 \times T_3$ of the two example graphs presented in table 2 and the example graph presented in table 5.

sizes are larger than the predicted ones on the upward slope of the peak, as the figure shows. Therefore, the predicted layers are larger on the falling slope. Additionally, the figure shows that the actually observed layer sizes reach their maximum at depth 14, whereas the estimation reaches its maximum at depth 15. Still, we can conclude that the estimates are fairly close to the actual values.

4.4 Experiments with state spaces of realistic systems

We are interested in whether the estimates using product state spaces give a better prediction for the sizes of state spaces that we encounter in practice. In figures 5, 6 and 7 the results are provided for the firewire protocol [7], the concurrent alternating bit protocol and the one place sliding window protocol [8]. On the x-axis the index of each layer is indicated. On the y-axis the estimated number of reachable states is depicted. Note that we use a logarithmic scale. The red squares denote the estimates assuming that the state space is a single random state space. The blue circles indicate the estimated number of states assuming that the state space consists of two random parallel systems. The numbers are calculated using the least square method using Matlab [1]. Using Mathematica, we were not able to obtain these results. To prevent the individual sizes of the components to become very different, we added a penalty $(N_1 - N_2)^2 10^{-6}$. The used scripts can be found in appendix A. The knowledge of the fan-out λ is not used. So, for the blue circles variables λ_1, λ_2 and N_1 and N_2 are estimated. Using these the size of the reachable state space is calculated and put in the figure.

It is obvious that the red boxes structurally underestimate the size of the reachable state space, where the blue circles do remarkably well. The figures in table 1 are different from those in figure 5 because they regard the size of the state space versus the size of the reachable state space. Furthermore, they have been obtained using different estimation methods. Still, they both suffer from the problem of structural underestimation.

Depths	1–5	2–6	3–7	4–8	5–9	6–10
$\mathbf{N_1}$	43.6	70.2	22.5	19.0	89.4	174.4
$\mathbf{N_2}$	543,056.2	$2.0 \cdot 10^6$	426.1	454.4	268.0	311.6
$\mathbf{N_3}$	320.3	584.2	452.4	451.4	372.5	332.3
$\mathbf{N}$	$7.6\ 10^9$	$82\ 10^9$	$4.3\ 10^6$	$3.9\ 10^6$	$8.9\ 10^6$	$18\ 10^6$
λ_1	2.28	2.75	1.91	1.85	1.97	1.81
λ_2	2.39	2.00	2.97	3.01	2.97	3.03

Depths	7–11	8–12	9–13	10–14	11–15	12–16
$\mathbf{N_1}$	114.4	176.5	113.2	*48.1*	46.9	48.4
$\mathbf{N_2}$	168.3	399.4	244.5	*33,191.4*	1292.9	213.1
$\mathbf{N_3}$	413.8	85.4	288.3	*1.1*	30.1	180.9
$\mathbf{N}$	$8.0\ 10^6$	$6.0\ 10^6$	$8.0\ 10^6$	$1.8\ 10^6$	$1.8\ 10^6$	$1.9\ 10^6$
λ_1	2.10	2.54	2.08	*1.49*	1.47	1.49
λ_2	3.18	3.28	3.52	*3.35*	3.00	2.59

Table 7: Results for solving for N_1, N_2, N_3, N, λ_1, and λ_2 in the product graph shown in table 6.

The observation that the obtained blue estimates are doing quite well supports the claim that 'real systems' behave as parallel non communicating state spaces. That the results contain variations is to be expected. If we look at 3D visualisations of the state space of the firewire protocol, then the two peaks at layer sizes 30 and 63 match very neatly with the disks in this 3D visualisation [6]. It is probably more remarkable that the red boxes are hardly influenced by these two peaks in layer sizes.

5 Estimating the presence of residual bugs

In this section we show that it matters very much if the system has a product graph structure if it comes to the effectiveness of testing for software bugs.

It is virtually impossible to test that there are no buggy states in a large random state space. However, if the state space consists of several parallel components, and the errors stem from those individual components, then testing for freedom of bugs is quite feasible.

We calculate how long a successful random walk through the state space must be to conclude with error α that there are no buggy states in the system. For this purpose we introduce two stochastic variables, namely M which represents the number of states in a walk without encountering a bug, and K which is the number of states with a bug. We assume that K is uniformly distributed where there are between 0 and n^p bugs. This upper bound on the number of bugs has little influence on the length of the random walk satisfying small α. So, we could as well assume that we know that there is a small upper bound on the number of states with bugs, without altering the results.

We compare a single random state space G of size n^p with the product of p state spaces $G_1, \ldots, G_p$, each of size n. We first calculate the probability that although there

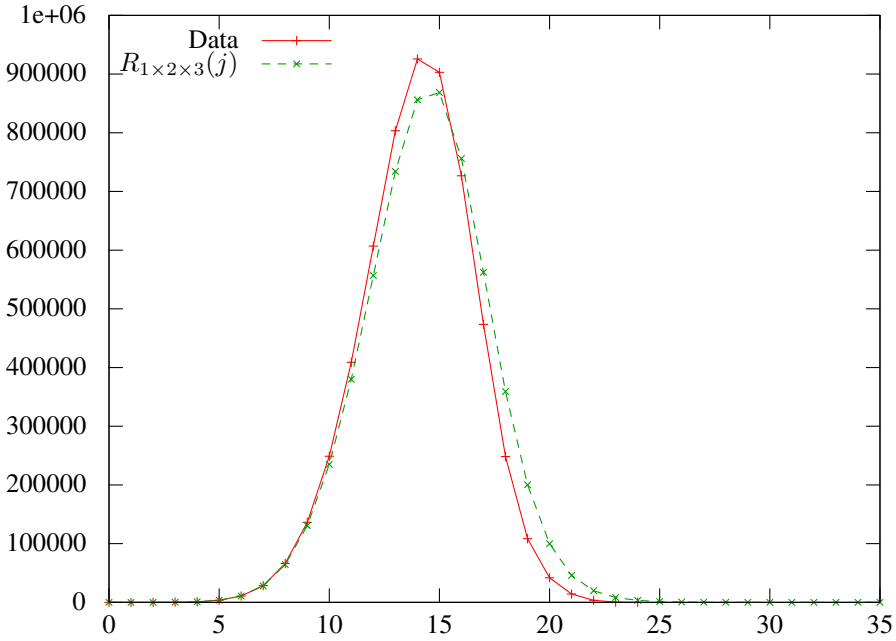

Figure 4: Comparison of observed layer sizes and predicted layer sizes for the product graph of table 6.

are states with bugs in the system, a random walk of size m does not encounter an error state in G. We want this probability to be smaller than α such that if we conclude that the system is free of bugs on the basis of a random walk of certain size, the probability that this is incorrect is smaller than α. Our estimation assumes that each time we visit a state its outgoing transitions go to random other states, possibly different than in a previous visit. Strictly spoken this is not correct as the outgoing transitions of a state are static, and when we revisit a state in a random traversal, the probability that the outgoing state is also already visited is higher than in our estimation. Taking the already visited states into account is a known difficult problem, and our slightly simplified model is already very useful to show heuristically the effect of the graph structure on testing. We compute

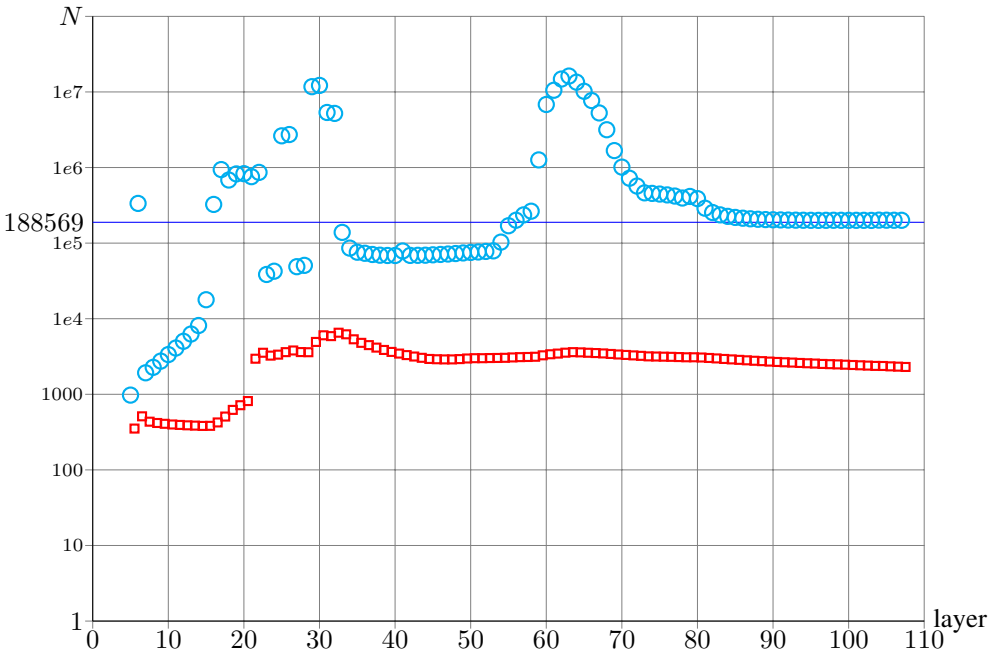

Figure 5: Estimates of the reachable state space of the firewire protocol

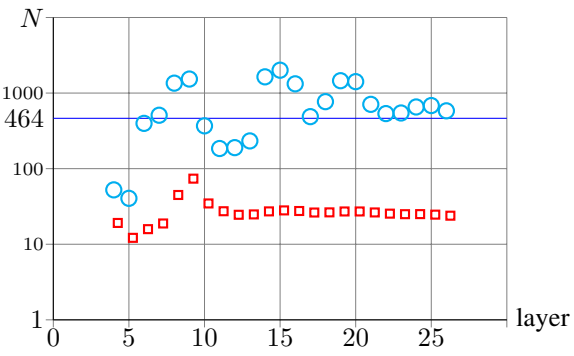

Figure 6: Estimates of the reachable state space of CABP

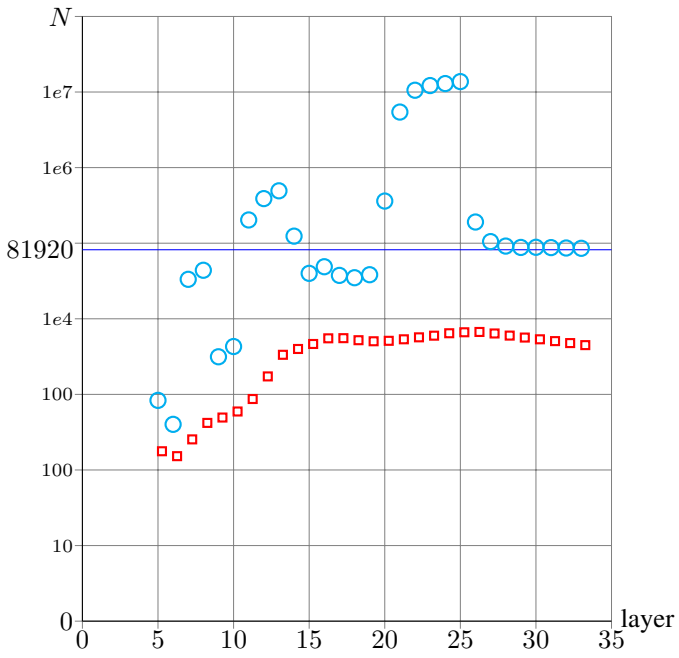

Figure 7: Estimates of the reachable state space of the one place sliding window protocol

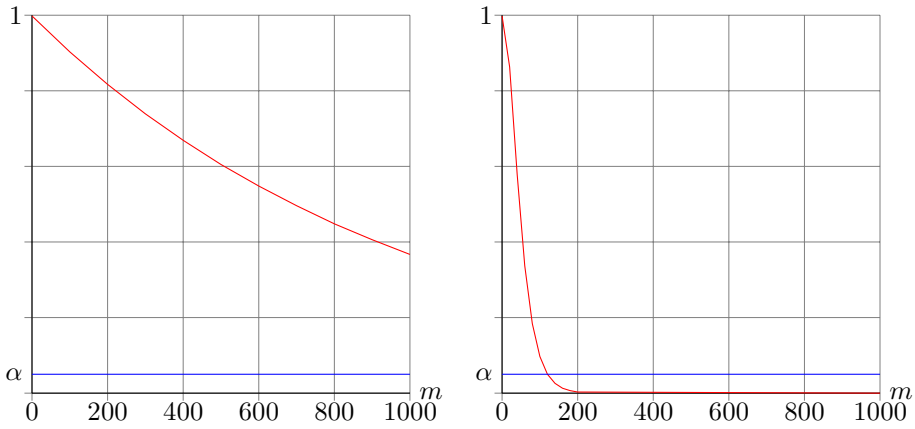

Figure 8: The probability of incorrectly declaring a system error free in G at the left and $G_1, \ldots, G_p$ at the right for $p = 3$ and $n = 10$.

$$
\begin{aligned}
\mathbb{P}[K>0|M=m] &= \frac{\sum_{k=1}^{n^p}\mathbb{P}[M=m|K=k]\,\mathbb{P}[K=k]}{\sum_{k=0}^{n^p}\mathbb{P}[M=m|K=k]\,\mathbb{P}[K=k]} \\
&= \frac{\sum_{k=1}^{n^p}(n^p-k)^m\,\mathbb{P}[K=k]}{\sum_{k=0}^{n^p}(n^p-k)^m\,\mathbb{P}[K=k]} \\
&= 1-\frac{n^{pm}\mathbb{P}[K=0]}{\sum_{k=0}^{n^p}(n^p-k)^m\mathbb{P}[K=k]} \\
&= 1-\frac{n^{pm}}{\sum_{k=0}^{n^p}(n^p-k)^m} < \alpha \qquad (18)
\end{aligned}
$$

At (18) we used that we assume that errors are uniformly distributed. So, $\mathbb{P}[K=0]$ equals $\mathbb{P}[K=k]$.

For the Cartesian product graph of G_1 to G_p we come to the following estimation where we assume that graph G_i has K_i error states, uniformly distributed from 0 to n. We write $\text{if}(c,x,y)$ for x if c holds, otherwise, it is y.

$$
\begin{aligned}
&\mathbb{P}[(\textstyle\sum_{i=1}^{p}K_i)>0|M=m] \\
&= \frac{\sum_{k_1=0}^{n}\cdots\sum_{k_p=0}^{n}\text{if}(\sum_{j=1}^{p}k_j{=}0,0,\mathbb{P}[M{=}m|\bigwedge_{i=1}^{p}K_i{=}k_i]\,\mathbb{P}[\bigwedge_{i=1}^{p}K_i{=}k_i])}{\sum_{k_1=0}^{n}\sum_{k_2=0}^{n}\cdots\sum_{k_p=0}^{n}\mathbb{P}[M{=}m|\bigwedge_{i=1}^{p}K_i{=}k_i]\,\mathbb{P}[\bigwedge_{i=1}^{p}K_i{=}k_i]} \\
&= \frac{\sum_{k_i=0}^{n}\cdots\sum_{k_p=0}^{n}\text{if}(\sum_{j=1}^{p}k_j=0,0,(1-\sum_{i=1}^{p}k_i/pn)^m\,\mathbb{P}[\bigwedge_{i=1}^{p}K_i=k_i])}{\sum_{k_1=0}^{n}\sum_{k_2=0}^{n}\cdots\sum_{k_p=0}^{n}(1-\sum_{i=1}^{p}k_i/pn)^m\,\mathbb{P}[\bigwedge_{i=1}^{p}K_i=k_i]} \\
&= 1-\frac{\mathbb{P}[\bigwedge_{i=1}^{p}K_i=0]}{\sum_{k_1=0}^{n}\sum_{k_2=0}^{n}\cdots\sum_{k_p=0}^{n}(1-\sum_{i=1}^{p}k_i/pn)^m\,\mathbb{P}[\bigwedge_{i=1}^{p}K_i=k_i]} \\
&= 1-\frac{1}{\sum_{k_1=0}^{n}\sum_{k_2=0}^{n}\cdots\sum_{k_p=0}^{n}(1-\sum_{i=1}^{p}k_i/pn)^m} < \alpha
\end{aligned}
$$

To obtain the third expression in the derivation above, we observe that a step that does not reach a state with a bug, does not hit such a state in any of the components. A component i has probability $(n-k_i)/n$ to avoid a state with a bug. Assuming that with equal probability each component can do a step, the probability to avoid a state with a bug is $\sum_{i=1}^{p}(n-k_i)/pn$. So, the probability to avoid m times a buggy states is $(1-\sum_{i=1}^{p}\frac{k_i}{np})^m$.

In figure 8 the estimates are depicted for a system with 1000 states versus three systems with 10 states. The difference is quite obvious. In the monolithic system a test run far longer than the number of states must be traversed to declare the system error free with $\alpha = 0.05$. With the parallel system a test run of far less than 200 steps is more than sufficient.

Notably, the number of required test runs only increases with the number of states per component. So, for four or more components with each 10 states test runs of

approximately 200 are also enough to obtain a certainty of 95% on the freedom of errors, whereas with the monolithic system test runs must have a length of over millions of steps to obtain the same effect.

6 Conclusions and Future Work

We first concluded that a standard random state space is not a good model for state spaces occurring in practice. As an alternative we proposed to use a number of parallel state spaces as a model of realistic state spaces. From experiments on the estimation of the sizes of state spaces we conclude that our model is much better.

As the state spaces from real applications are non random, it is hard to come to a definitive conclusion that our model is good. Real state spaces have typical features, like a long sequential initialisation phase, after which behaviour becomes parallel. It is impossible to capture such behaviour exactly. But still, a decent random approximation of realistic state spaces may help to shed light on phenomena that are not really understood well, and even come up with decent bounds on for instance the number of residual bugs in a system.

Obtaining the numerical estimates in this paper was far from trivial. Initially, we used Mathematica and later Matlab to obtain the results. When estimating the sizes of parallel state spaces, we assumed that they consisted of two components. It is interesting to redo the experiments with more components, and it is also challenging to estimate the number of parallel components, i.e., the index of parallelism, of a given realistic system, based on experimental data.

When testing software, experience is that some bugs are impossible to find, while many others pop up immediately. Applying the new model to bug detection explains that testing is an effective technique when bugs stem from single components, and we can calculate the number of required random tests to guarantee the absence of certain types of bugs with a given uncertainty. If a bug occurs when n components are all in specific states, it might be hard to locate. It appears that therefore, it makes sense to divide bugs in classes. A class I bug stems from one component, and is relatively easy to detect. A class n bug is the result of n parallel components being in a specific state. Such bugs are harder to find for increasing n.

We present some evidence that our model is a better approximation of real systems than more standard monolithic random graphs. The interesting question is whether our proposed model is adequate to study typical phenomena in the behaviour of computerised systems, or that other types of random models are even more adequate. The only way to know this is through further investigation.

Acknowledgments

We would like to thank (in alphabetical order) Jesse Goodman for his input on this paper's topic and Wil Kortsmit for his assistance with the Mathematica implementation. This research was in part supported by NWO and Agentschap NL.

A Matlab scripts and input data

This appendix contains a few Matlab scripts that have been run on Matlab version R2013a to obtain the numbers in figures 5, 6 and 7. The function `P_single` estimates the size of a state space assuming that it is monolithic random. It gets a layer list which contains the measured sizes of the layers. At the end of this appendix the used layer sizes are provided. The value `lambda` is an initial value of the estimate of the fan out, and it has no bearing on the actual value of the estimated fan out. The variables `from` and `to` indicate which measured layer sizes must be used to estimate the size of the state space.

```
function [y] = P_single(layer_size_list,lambda,from,to)
options=optimset('MaxFunEvals',100000,'MaxIter',10000);
starting_point=[100,lambda];
[x,fval,exitflag,output]=fminsearch(@(x)
     kwad_sum(layer_size_list,x(1),x(2),from,to),
                starting_point,options);
x(1)   % Number of estimated states of the component.
x(2)   % Estimated lambda
no_states=fzero(@(s)s-x(1)*(1-((x(1)-1)/x(1))^(1+x(2)*s)),x(1));
y=no_states;  % Total number of reachable states.
end

function [y] = kwad_sum(l,Nm,lambda,from,to)
sum=0;
for i=from:to
    sum=sum+(Prod(i-1,Nm,lambda)-l(i))^2;
end
y=sum;
end

function [y] = Prod(j,Nm,lambda)
global Rcache;
global Scache;
Rcache(2,j+1)=0;
Scache(2,j+1)=0;
sum=0;
fill_cache(1,j+1,Nm,lambda);
for k=0:j
    v1=Rcache(1,k+1);
    sum=sum+v1;
end
y=sum;

end

function fill_cache(n,j,Nm,lambda)
global Rcache;
global Scache;
```

```
for i=1:j+1
    Rcache(n,i)=R(n,i-1,Nm,lambda);
    Scache(n,i)=S(n,i-1,Nm,lambda);
end
end

function [ y ] = R(n,j,Nm,lambda )
global Rcache;
global Scache;
  if (j==0)
      y=1;
  else
      y=(Nm-Scache(n,j))*(1-((Nm-1)/Nm)^(lambda*Rcache(n,j)));
  end
end

function [ y ] = S(n,j,Nm,lambda )
global Rcache;
global Scache;

  if (j==0)
      y=1;
  else
      y=Rcache(n,j+1)+Scache(n,j);
  end

end
```

The function `P` estimates the size of the state space assuming it consists of two parallel components. The arguments are exactly the same as for `P_single`.

```
function [y] = P(layer_size_list,lambda,from,to)
options=optimset('MaxFunEvals',10000000,'MaxIter',1000000);
starting_point=[100,100,lambda/2,lambda/2];

[x,fval,exitflag,output]=
    fminsearch(@(x)
        kwad_sum(layer_size_list,x(1),x(2),x(3),x(4),from,to),
        starting_point,options);
states_first_component=x(1)  % Number of states of component 1.
states_second_component=x(2) % Number of states of component 2.
lambda1=x(3)   % Estimated lambda1
lambda2=x(4)   % Estimated lambda2
no_states=x(1)*x(2); % Total number of states
                     % including those that are not reachable.
% Total number of reachable states of both components.
no_states1=fzero(@(s)max(s,0)-x(1)*(1-((x(1)-1)/x(1))^
         (1+x(3)*max(s,0))),1+real(x(1)/10));
no_states2=fzero(@(s)max(0,s)-x(2)*(1-((x(2)-1)/x(2))^
         (1+x(4)*max(s,0))),1+real(x(2)/10));
total_number_of_reachable_states=no_states1*no_states2;
```

```
y=total_number_of_reachable_states;
end

function [y] = kwad_sum(l,Nm1,Nm2,lambda1,lambda2,from,to)
% The following is added to let Nm1 and Nm2 not differ too much.
sum=1e-6*(Nm1-Nm2)^2;
for i=from:to
    sum=sum+(Prod(i-1,Nm1,Nm2,lambda1,lambda2)-l(i))^2;
end
y=sum;
end

function [y] = Prod(j,Nm1,Nm2,lambda1,lambda2)
global Rcache;
global Scache;
Rcache(2,j+1)=0;
Scache(2,j+1)=0;
sum=0;
fill_cache(1,j+1,Nm1,lambda1);
fill_cache(2,j+1,Nm2,lambda2);
for k=0:j
    v1=Rcache(1,k+1);
    v2=Rcache(2,j-k+1);
    sum=sum+v1*v2;
end
y=sum;

end

function fill_cache(n,j,Nm,lambda)
global Rcache;
global Scache;

for i=1:j+1
    Rcache(n,i)=real(R(n,i-1,Nm,lambda));
    Scache(n,i)=real(S(n,i-1,Nm,lambda));
end
end

function [ y ] = R(n,j,Nm,lambda )
global Rcache;
global Scache;
  if (j==0)
      y=1;
  else
      y=(Nm-Scache(n,j))*(1-((Nm-1)/Nm)^(lambda*Rcache(n,j)));
  end
end

function [ y ] = S(n,j,Nm,lambda )
```

```
global Rcache;
global Scache;

  if (j==0)
      y=1;
  else
      y=Rcache(n,j+1)+Scache(n,j);
  end

end
```

The used layer sizes are

```
layerlist_firewire=[1, 24, 168, 312, 312, 600, 312, 312, 312,
312, 312, 312, 312, 312, 348, 816, 1280, 1664, 1672, 1796, 2184,
2776, 2424, 2844, 3292, 3432, 2802, 3085, 6709, 5840, 4827,
6230, 4866, 2107, 1545, 1794, 846, 546, 553, 636, 624, 620, 572,
688, 2090, 2233, 2621, 3193, 3893, 3284, 2640, 3088, 2736, 3156,
3604, 3744, 3114, 3409, 7189, 6464, 5435, 6338, 5130, 2371, 1665,
1806, 1458, 1146, 865, 952, 987, 935, 586, 2176, 2101, 2003, 1892,
1781, 2485, 1776, 600, 312, 312, 312, 312, 312, 312, 324, 480,
624, 624, 624, 624, 624, 624, 648, 936, 912, 624, 624, 624, 588,
300, 1752, 288, 288, 288]

layerlist_cabp=[1, 3, 6, 10, 10, 14, 16, 24, 30, 24, 17, 15, 23,
33, 28, 22, 16, 24, 30, 24, 16, 12, 17, 23, 18, 8]

layerlist_sliding=layerlist=[1, 6, 15, 38, 87, 122, 198, 308, 397,
508, 732, 1148, 1760, 2372, 3100, 3976, 4336, 4172, 4166, 4612,
5298, 5956, 6238, 6656, 7173, 6190, 4049, 2534, 2083, 1894, 1254,
468, 73]
```

As an example, the predicted reachable state space size for firewire, after having observed 30 layers assuming that it consists out of 2 parallel components, can be obtained by

```
P(layerlist_firewire,3,1,30)
```

And the same prediction, assuming that it is one monolithic state space is obtained by

```
P_single(layerlist_firewire,3,1,30)
```

References

[1] S. Attaway. Matlab, third edition: a practical introduction to programming and problem solving. Third Edition. Butterworth-Heinemann publishers, Elsevier, 2013.

[2] B. Bollobás. Random Graphs. Cambridge University Press, 2001.

[3] N.J. Dingle and W.J. Knottenbelt. State-Space Size Estimation By Least-Squares Fitting. In *Proceedings of the 24th UK Performance Engineering Workshop (UKPEW'08)*, pages 347–357, 2008.

[4] T.A.N. Engels, J.F. Groote, M.J. van Weerdenburg and T.A.C. Willemse. Search algorithms for automated validation. Journal of Logic and Algebraic Programming 78(4), 274-287, 2009.

[5] J.F. Groote et al. The mCRL2 toolset. In *Proceedings of the International Workshop on Advanced Software Development Tools and Techniques (WASDeTT'08)*, 2008.

[6] J.F. Groote and F.J.J. van Ham. Interactive visualization of large state spaces. International Journal on Software Tools for Technology Transfer 8:77-91, 2006.

[7] S.P. Luttik. Description and formal specification of the Link Layer of P1394. In: Ignac Lovrek, editor, Proceedings of the 2nd International Workshop on Applied Formal Methods in System Design, University of Zagreb, Croatia, pp. 43-56, June 1997.

[8] S. Mauw and G.J. Veltink (editors). Algebraic specification of communication protocols. Cambridge tracts in theoretical computer science 36. Cambridge University Press. 1993.

[9] J.F. Watson III and A.A. Desrochers. State-Space Size Estimation of Petri Nets: A Bottom-Up Perspective. *IEEE Transactions on Robotics and Automation*, 10(4):555–561, 1994.

[10] Wolfram Research Inc. Mathematica 8.0.0.0. See also `http://www.wolfram.com`.

[11] F.C. Lincoln. Calculating Waterfowl Abundance on the Basis of Banding Returns. *United States Department of Agriculture Circular*, 118:1–4, 1930.

[12] R. Pelánek and P. Šimeček. Estimating State Space Parameters. Technical Report FIMU-RS-2008-01, Faculty of Informatics, Masaryk University, 2008.

Science Reports **Department of Mathematics and Computer Science**
Technische Universiteit Eindhoven

If you want to receive reports, send an email to: wsinsan@tue.nl (we cannot guarantee the availability of the requested reports).

In this series appeared (from 2012):

12/01	S. Cranen	Model checking the FlexRay startup phase
12/02	U. Khadim and P.J.L. Cuijpers	Appendix C / G of the paper: Repairing Time-Determinism in the Process Algebra for Hybrid Systems ACP
12/03	M.M.H.P. van den Heuvel, P.J.L. Cuijpers, J.J. Lukkien and N.W. Fisher	Revised budget allocations for fixed-priority-scheduled periodic resources
12/04	Ammar Osaiweran, Tom Fransen, Jan Friso Groote and Bart van Rijnsoever	Experience Report on Designing and Developing Control Components using Formal Methods
12/05	Sjoerd Cranen, Jeroen J.A. Keiren and Tim A.C. Willemse	A cure for stuttering parity games
12/06	A.P. van der Meer	CIF MSOS type system
12/07	Dirk Fahland and Robert Prüfer	Data and Abstraction for Scenario-Based Modeling with Petri Nets
12/08	Luc Engelen and Anton Wijs	Checking Property Preservation of Refining Transformations for Model-Driven Development
12/09	M.M.H.P. van den Heuvel, M. Behnam, R.J. Bril, J.J. Lukkien and T. Nolte	Opaque analysis for resource-sharing components in hierarchical real-time systems - extended version –
12/10	Milosh Stolikj, Pieter J. L. Cuijpers and Johan J. Lukkien	Efficient reprogramming of sensor networks using incremental updates and data compression
12/11	John Businge, Alexander Serebrenik and Mark van den Brand	Survival of Eclipse Third-party Plug-ins
12/12	Jeroen J.A. Keiren and Martijn D. Klabbers	Modelling and verifying IEEE Std 11073-20601 session setup using mCRL2
12/13	Ammar Osaiweran, Jan Friso Groote, Mathijs Schuts, Jozef Hooman and Bart van Rijnsoever	Evaluating the Effect of Formal Techniques in Industry
12/14	Ammar Osaiweran, Mathijs Schuts, and Jozef Hooman	Incorporating Formal Techniques into Industrial Practice
13/01	S. Cranen, M.W. Gazda, J.W. Wesselink and T.A.C. Willemse	Abstraction in Parameterised Boolean Equation Systems
13/02	Neda Noroozi, Mohammad Reza Mousavi and Tim A.C. Willemse	Decomposability in Formal Conformance Testing
13/03	D. Bera, K.M. van Hee and N. Sidorova	Discrete Timed Petri nets
13/04	A. Kota Gopalakrishna, T. Ozcelebi, A. Liotta and J.J. Lukkien	Relevance as a Metric for Evaluating Machine Learning Algorithms
13/05	T. Ozcelebi, A. Weffers-Albu and J.J. Lukkien	Proceedings of the 2012 Workshop on Ambient Intelligence Infrastructures (WAmIi)
13/06	Lotfi ben Othmane, Pelin Angin, Harold Weffers and Bharat Bhargava	Extending the Agile Development Process to Develop Acceptably Secure Software
13/07	R.H. Mak	Resource-aware Life Cycle Models for Service-oriented Applications managed by a Component Framework
13/08	Mark van den Brand and Jan Friso Groote	Software Engineering: Redundancy is Key
13/09	P.J.L. Cuijpers	Prefix Orders as a General Model of Dynamics

14/01	Jan Friso Groote, Remco van der Hofstad and Matthias Raffelsieper	On the Random Structure of Behavioural Transition Systems