Extracting Passage Times from PEPA models with the HYDRA Tool: a Case Study

Jeremy T. Bradley[1] Nicholas J. Dingle[1] Stephen T. Gilmore[2]
William J. Knottenbelt[1]

[1] Department of Computing, Imperial College London
180 Queen's Gate, London SW7 2BZ, United Kingdom.
{jb,njd200,wjk}@doc.ic.ac.uk
[2] Laboratory for the Foundations of Computer Science,
University of Edinburgh, Edinburgh EH9 3JZ, United Kingdom.
stg@dcs.ed.ac.uk

Abstract. Passage time densities and quantiles are important performance metrics which are increasingly used in specifying service level agreements (SLAs) and benchmarks. PEPA is a popular stochastic process algebra and a powerful formalism for describing performance models of communication and computer systems. We present a case study passage time analysis of an 82,944 state PEPA model using the HYDRA tool. HYDRA specialises in passage time analysis of large Markov systems based on stochastic Petri nets. By using the new Imperial PEPA compiler (ipc), we can construct a HYDRA model from a PEPA model and obtain passage time densities based on the original PEPA description.

1 Introduction

Traditionally, performance analysis of concurrent systems has focused on steady-state analysis—that is, on calculating the long-run probability that a system will be in each of its reachable states. This is adequate to predict standard resource-based measures, such as throughput and utilisation, but is inadequate to answer questions about transient state distributions, such as "What is the probability that a router has 10 packets in its buffer 20 seconds after startup?", or questions about passage time quantiles, such as "What is the probability that a packet will pass through a router in under 5 milliseconds?". The latter question is particularly important since passage time quantiles are increasingly specified as key quality-of-service metrics in benchmarks and Service Level Agreements.

PEPA [1, 2] is a popular Markovian process algebra for specifying compositional performance models. PEPA models reduce to an underlying Markov chain, which can be analysed using appropriate Markovian techniques. While PEPA Workbench [3, 4] can produce both steady-state and transient results, we concentrate here on an example calculation of a passage time from a model expressed in PEPA.

The HYpergraph-based Distributed Response-time Analyser (HYDRA) [5] is a parallel tool that uses an efficient uniformization-based technique to compute passage time densities and quantiles in Markov chains with large state spaces of the order of 10^7 states. It is an extension of DNAmaca [6, 7], a tool for the steady-state analysis of large low-level Markov [8] and semi-Markov [9] chains. By using state-of-the-art hypergraph partitioning techniques HYDRA yields excellent scalability on distributed memory parallel computers and is able to utilise effectively the compute power and RAM provided by a network of workstations.

In this paper, we present a case study showing how passage time quantities can be calculated from PEPA models using the new *Imperial PEPA compiler* (ipc) and the HYDRA tool. ipc translates a PEPA specification at the component level (as opposed to the global state-space level) to a stochastically identical HYDRA specification. We then use HYDRA to extract the passage time densities and quantiles of interest. We demonstrate the applicability of our approach by the analysis of an 82, 944 state PEPA model of an active badge system of people moving along a corridor of rooms (extended from the original model presented in [10]).

2 PEPA

PEPA is a parsimonious stochastic process algebra that can concisely describe compositional models. These models consist of components whose actions have exponentially distributed rates. The syntax of a PEPA component P is represented by:

$$P ::= (a, \lambda).P \big| P + P \big| P \underset{S}{\bowtie} P \big| P/L \big| A \tag{1}$$

where:

$(a, \lambda).P$ is a prefix operation defining a component which performs an activity of type a, with a duration exponentially distributed with rate λ, before evolving into P.

$P_1 + P_2$ is a choice operation. A race is entered into between components P_1 and P_2. If P_1 evolves first then any behaviour of P_2 is discarded and vice-versa.

$P_1 \underset{S}{\bowtie} P_2$ is the cooperation operator. P_1 and P_2 run in parallel and synchronise over the set of actions in the set S. If P_1 is to evolve with an action $a \in S$, then it must first wait for P_2 to reach a point where it is capable of producing an a action (and *vice versa*). The two components then jointly execute an a-action with a rate that reflects the slower of the two components (usually the minimum of the two individual a-rates).

P/L is a hiding operator whereby actions in the set L that emanate from the component P are rewritten as silent τ actions (with the same appropriate delays). The actions in L can no longer be used in cooperation with other components.

A is a constant label which allows, amongst other things, for recursive definitions to be constructed.

3 HYDRA Model Description

The HYDRA model description language is based on the DNAmaca interface language detailed in [6] which allows for the high-level description of Markovian models such as stochastic Petri nets and queueing networks.

We provide an excerpt from a sample HYDRA specification below, showing:

- constant declarations for MM to 45, NN to 5 and CC to 175
- type declaration of the state or marking vector elements p_1 to p_7 to `short`
- configuration of the initial marking, setting p_3 to MM and p_5 to NN
- a sample definition of a transition, t_5, where:
  - `\condition` sets the firing condition that must be met for the marking to be enabled
  - `\action` denotes how the global marking will change once the transition fires
  - `\rate` specifies the exponential rate parameter for the delay incurred when firing the transition
  - `\priority` allows for priority selection of the highest priority transitions

```
\model{
  \constant{MM}{45}
  \constant{NN}{5}
  \constant{CC}{175}
  \statevector{
    \type{short}{ p1, p2, p3, p4, p5, p6, p7 }
  }
  \initial{
    p1 = CC; p2 = 0; p3 = MM; p4 = 0; p5 = NN; p6 = 0; p7 = 0;
  }
  ...
  \transition{t5}{
    \condition{p7 > MM-1}
    \action{
      next->p3 = p3 + MM;
      next->p7 = p7 - MM;
    }
    \rate{3.2}
    \priority{1}
  }
  ...
}
\solution{
  \method{sor}
}
\passage{
  \sourcecondition{ (p1 == CC && p3 == MM && p5 == NN) }
  \targetcondition{ (p2 == CC) }
  \t_start{50.0}
  \t_stop{200.0}
  \t_step{5.0}
}
```

After the model description comes the specification of analysis to be performed on the model. The `\solution` directive identifies which numerical method should be used for steady-state solution. Of particular interest to this paper is the `\passage` directive, however, which specifies a passage time density function using the start and end marking(s) of the passage. In this case, the passage can start from the state where $p_1 = CC$, $p_3 = MM$ and $p_5 = NN$ but can terminate in any state where $p_2 = NN$. If there are many source states to a passage, then the steady-state distribution is used to weight the different possible passages, to give an overall passage distribution at equilibrium [11]. The resulting density function is produced by sampling at intervals specified by `\t_step` between the time points, `\t_start` and `\t_stop`.

4 Tool Theory and Architecture

4.1 Technical Summary

PEPA models reduce to an underlying continuous-time Markov chain (CTMC), so we consider a CTMC with rate matrix $Q = q_{ij}$. Solving the linear system $\pi Q = 0$ subject to $\sum \pi_i = 1$, gives us the steady state vector, π. HYDRA calculates passage time densities from many source states $\boldsymbol{i}$ to many target states $\boldsymbol{j}$ by means of an efficient uniformization-based analysis.

Uniformization [12, 13] transforms a CTMC into one in which all states have the same mean holding time $1/q$, by allowing *invisible* transitions from a state to itself. This is equivalent to a discrete-time Markov chain (DTMC), after normalising the rows, together with an associated Poisson process of rate q. The one-step DTMC transition matrix, P, is given by:

$$P = Q/q + I \tag{2}$$

where $q > \max_i |q_{ii}|$ (to ensure that the DTMC is aperiodic) and I is the identity matrix.

While uniformization is normally used for transient analysis, it can also be employed for the calculation of passage time densities and quantiles [14, 15]. We add an extra, absorbing state to our uniformized chain, which is the sole successor state for all target states (thus ensuring we calculate the *first* passage time density). We denote by P', the one-step transition matrix of the modified, uniformized chain. Remembering that the time taken to traverse a path with n hops in this chain will have an Erlang distribution with parameters n and q, the density of the time taken to pass from a set of source states $\boldsymbol{i}$ into a set of target states $\boldsymbol{j}$ is given by:

$$f_{\boldsymbol{ij}}(t) = \sum_{n=1}^{\infty} \frac{q^n t^{n-1} e^{-qt}}{(n-1)!} \sum_{k \in \boldsymbol{j}} \pi_k^{(n)} \tag{3}$$

where

$$\pi^{(n+1)} = \pi^{(n)} P' \qquad : \text{for } n \geq 0$$

with

$$\pi_k^{(0)} = \begin{cases} 0 & : \text{for } k \notin \boldsymbol{i} \\ \pi_k / \sum_{j \in \boldsymbol{i}} \pi_j & : \text{for } k \in \boldsymbol{i} \end{cases} \tag{4}$$

and in which $\boldsymbol{\pi}$ is any non-zero solution to $\pi = \pi P$. The corresponding passage time cumulative distribution function is given by:

$$F_{\boldsymbol{ij}}(t) = \sum_{n=1}^{\infty} \left\{ \left(1 - e^{-qt} \sum_{k=0}^{n-1} \frac{(qt)^k}{k!} \right) \sum_{k \in \boldsymbol{j}} \pi_k^{(n)} \right\}. \tag{5}$$

Truncation is employed to approximate the infinite sum in Eq. (3) (and Eq. (5)), terminating the calculation when the Erlang term drops below a specified threshold value. Concurrently, when the convergence criterion

$$\frac{||\pi^{(n+1)} - \pi^{(n)}||_\infty}{||\pi^{(n)}||_\infty} < \epsilon \tag{6}$$

is met, for given tolerance ϵ, the steady state probabilities of P' are considered to have been obtained with sufficient accuracy and no further multiplications with P' are performed.

4.2 Tool Pipeline

The overall pipeline for the passage time analysis of a PEPA model goes as follows:

1. `ipc` compilation of `.pepa` file to HYDRA's native `.mod` format:
   (a) PEPA normal form translation
   (b) Component state space exploration
   (c) HYDRA component linking for shared transitions and `.mod` file generation
2. Construct passage start and termination conditions in terms of HYDRA marking specification.
3. HYDRA analysis:
   (a) solves for steady state vector using selected algorithm e.g. SOR, CGS, Gauss-Seidel
   (b) solves for the required passage time density or quantile according to Section 4.1

We show how meaningful passage time specifications from PEPA models can be expressed in HYDRA in the next section.

5 Worked Example

5.1 General Active Badge Model

In the original active badge model, described in [10], there are 4 rooms on a corridor, all installed with active badge sensors, and a single person who can move from one room to an adjacent room. The sensors are linked to a database which records which sensor has been activated last. In the model of Fig. 1, we have M people in N rooms with sensors and a database that can be in one of N states. To maintain a reasonable state space, this is a simple database which does not attempt to keep track of every individual's location, rather it remembers the last movement that was made by any person in the system.

In the model below, Person_i represents a person in room i, Sensor_i is the sensor in room i and Dbase_i is the state of the database. A person in room i can either move to room $i-1$ or $i+1$ or, if they remain there long enough, set off the sensor in room i, which registers its activation with the database.

The first thing to note about such a model is how fast the state space can grow. With M people in N rooms, we already have N^M states just from the different configurations of people in rooms. Then there are 2^N sensor configurations and finally N states that the database can be in, giving us a total of $2^N N^{M+1}$ states. For as few as 3 people and 6 rooms, the example we use, we have a global state space of $82,944$ states.

$$\begin{aligned}
\text{Person}_1 &= (reg_1, r).\text{Person}_1 + (move_2, m).\text{Person}_2 \\
\text{Person}_i &= (move_{i-1}, m).\text{Person}_{i-1} + (reg_i, r).\text{Person}_i \\
&\quad + (move_{i+1}, m).\text{Person}_{i+1} \\
&\quad\ : 1 < i < N \\
\text{Person}_N &= (move_{N-1}, m).\text{Person}_{N-1} + (reg_N, r).\text{Person}_N \\
\\
\text{Sensor}_i &= (reg_i, \top).(rep_i, s).\text{Sensor}_i \quad : 1 \le i \le N \\
\\
\text{Dbase}_i &= \textstyle\sum_{j=1}^{N} (rep_j, \top).\text{Dbase}_j \quad : 1 \le i \le N \\
\\
\text{Sys} &= \textstyle\prod_{j=1}^{M} \text{Person}_1 \underset{Reg}{\bowtie} \prod_{j=1}^{N} \text{Sensor}_j \underset{Rep}{\bowtie} \text{Dbase}_1
\end{aligned}$$

where $Reg = \{\text{reg}_i \mid 1 \le i \le N\}$ and $Rep = \{\text{rep}_i \mid 1 \le i \le N\}$

Fig. 1. The PEPA description for the generalised active badge model with N rooms and M people.

5.2 Results

We include two passages from the active badge system with 3 people and 6 possible rooms (the `.pepa` file for this is given in Appendix A). As the model of Fig. 1 tells us, all 6 people start in room 1 and move out from there.

The place labels reflect the way `ipc` translates repeat components, in this case 3 copies of Person$_1$ in parallel, into independent HYDRA subnets. The first person, Person$_i$, gets converted to `Person`i, the second to `Person`i`_1` and the third to `Person`i`_2`, for each room $1 \le i \le 6$. So asking if person 2 is in room 4 in HYDRA terms is (`Person4_1 == 1`).

Fig. 2 shows the density function for the passage representing how long it takes for all 3 people to be together in room 6 for the first time. So, using the above reasoning, this was achieved using the HYDRA passage specification:

```
\passage{
  \sourcecondition{ (Person1 == 1)
       && (Person1_1 == 1) && (Person1_2 == 1) }
  \targetcondition{ (Person6 == 1)
       && (Person6_1 == 1) && (Person6_2 == 1) }
}
```

It is interesting to observe that it is virtually impossible for all 3 people to end up in room 6, which requires 6 successive *move* transitions from all 3 people for it to happen at the earliest opportunity, until at least 10 time units have elapsed.

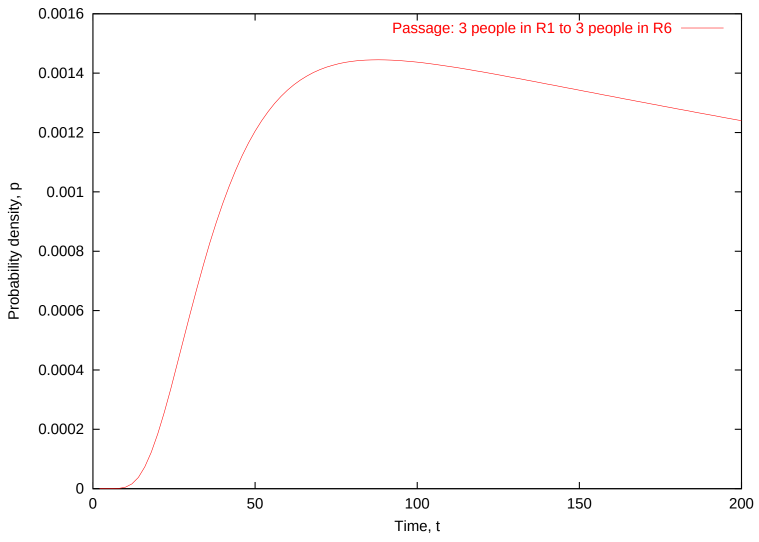

Fig. 2. The passage time density for 3 people starting in room 1 ending up all in room 6.

After that time, very low probabilities are registered and the distribution clearly has a very heavy tail.

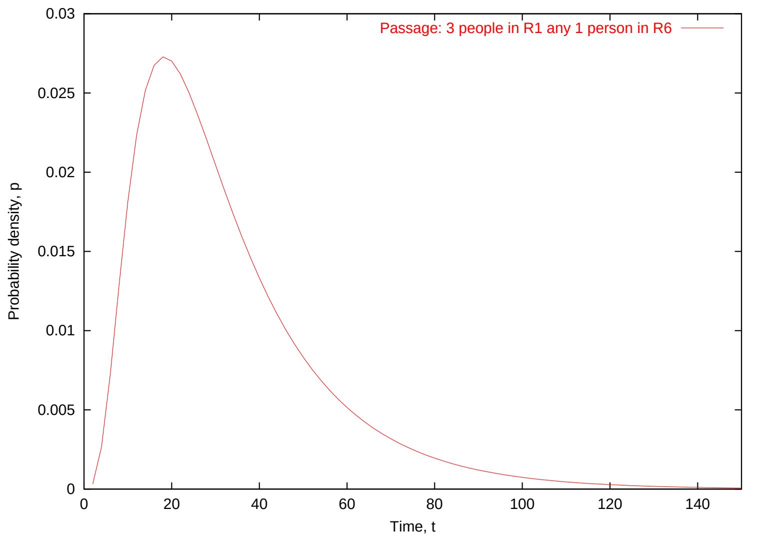

Fig. 3. The passage time density for 3 people starting in room 1 ending up with any one or more of them in room 6.

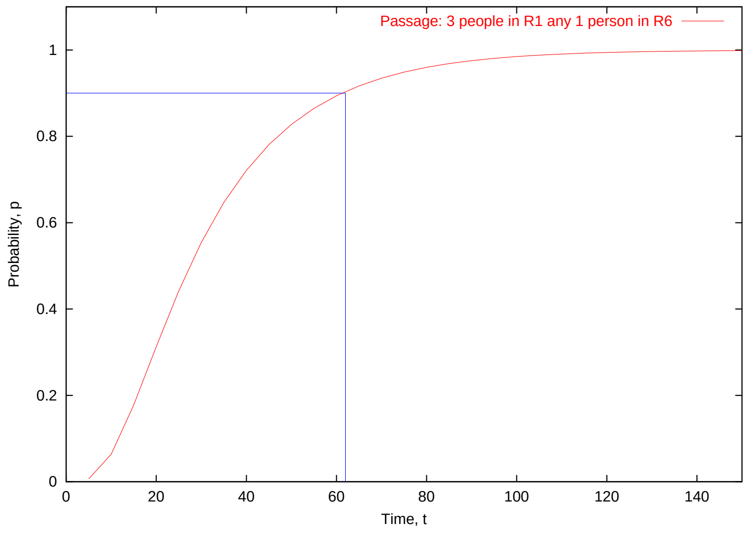

Fig. 4. The cumulative passage time distribution function for 3 people starting in room 1 ending up with any one or more of them in room 6.

The second passage of Fig. 3 shows an equivalent passage time density from the same start point to a terminating condition of at least one person of the three entering room 6. This is achieved similarly with a HYDRA passage definition of:

```
\passage{
  \sourcecondition{ (Person1 == 1) &&
       (Person1_1 == 1) && (Person1_2 == 1) }
  \targetcondition{ (Person6 == 1) ||
       (Person6_1 == 1) || (Person6_2 == 1) }
}
```

where the logical ANDs of the first target specification are replaced by logical ORs. The resulting passage is much less heavy tailed, as this time only a single person has to make it to room 6 before the passage ends.

From these densities, it is a simple matter to construct cumulative distribution functions (the integral of the density function) and obtain quantiles, e.g. the probability that 3 people all reach room 6 by time, $t = 150$. Fig. 4 shows the cumulative distribution function (cdf) corresponding to the passage time density of Fig. 3. From this cdf, we can ascertain, for example, that there is a 90% probability that at least one person will have reached room 6 by time $t = 62$.

6 Conclusion

In this paper, we presented a PEPA model of an active badge system and used it to demonstrate passage time analysis. Using ipc, we translate the PEPA model to a stochastically identical HYDRA model. From there, we make use of the HYDRA tool to produce passage time quantities.

Although the ipc tool automatically generates the main body of the HYDRA model description from the corresponding PEPA description, it is not yet possible to generate passage time definitions from PEPA directly. As future work, we therefore plan to relate the desired passage time quantity to the action behaviour of the PEPA model and generate the source and target state descriptions for HYDRA automatically.

References

1. J. Hillston, "Compositional Markovian modelling using a process algebra," in *Proceedings of the 2nd International Workshop on Numerical Solution of Markov Chains*, Kluwer Academic Press, Raleigh, January 1995.
2. J. Hillston, *A Compositional Approach to Performance Modelling*, vol. 12 of *Distinguished Dissertations in Computer Science.* Cambridge University Press, 1996. ISBN 0 521 57189 8.
3. S. Gilmore and J. Hillston, "The PEPA workbench: A tool to support a process algebra-based approach to performance modelling," in *Proceedings of the 7th International Conference on Modelling Techniques and Tools for Computer Performance Evaluation* (G. Haring and G. Kotsis, eds.), vol. 794 of *Lecture Notes in Computer Science*, pp. 353–368, Springer-Verlag, Vienna, May 1994.
4. G. Clark, S. Gilmore, J. Hillston, and N. Thomas, "Experiences with the PEPA performance modelling tools," in *UKPEW'98, Proceedings of the 14th UK Performance Engineering Workshop*, Edinburgh, July 1998.
5. N. J. Dingle, W. J. Knottenbelt, and P. G. Harrison, "HYDRA: HYpergraph-based Distributed Response-time Analyser," in *PDPTA'03, Proceedings of the 2003 International Conference on Parallel and Distributed Processing Techniques and Applications* (H. R. Arabnia and Y. Man, eds.), vol. 1, (Las Vegas, NV), pp. 215–219, June 2003.
6. W. J. Knottenbelt, "Generalised Markovian analysis of timed transitions systems," MSc thesis, University of Cape Town, South Africa, July 1996.
7. W. J. Knottenbelt, *Parallel Performance Analysis of Large Markov Models.* PhD thesis, Imperial College, London, United Kingdom, February 2000.
8. W. J. Knottenbelt and P. G. Harrison, "Distributed disk-based solution techniques for large Markov models," in *NSMC'99, Proceedings of the 3rd Intl. Conference on the Numerical Solution of Markov Chains*, (Zaragoza), pp. 58–75, September 1999.
9. J. T. Bradley, N. J. Dingle, W. J. Knottenbelt, and P. G. Harrison, "Performance queries on semi-Markov stochastic Petri nets with an extended Continuous Stochastic Logic," in *PNPM'03, Proceedings of Petri Nets and Performance Models* (G. Ciardo and W. Sanders, eds.), (University of Illinois at Urbana-Champaign), pp. 62–71, IEEE Computer Society, September 2003. (To appear).

10. S. Gilmore, J. Hillston, and G. Clark, “Specifying performance measures for PEPA,” in *Proceedings of the 5th International AMAST Workshop on Real-Time and Probabilistic Systems*, vol. 1601 of *Lecture Notes in Computer Science*, (Bamberg), pp. 211–227, Springer-Verlag, 1999.
11. P. G. Harrison and W. J. Knottenbelt, “Passage-time distributions in large Markov chains,” in *Proceedings of ACM SIGMETRICS 2002* (M. Martonosi and E. d. S. e Silva, eds.), pp. 77–85, Marina Del Rey, USA, June 2002.
12. W. Grassman, “Means and variances of time averages in Markovian environments,” *European Journal of Operational Research*, vol. 31, no. 1, pp. 132–139, 1987.
13. A. Reibman and K. Trivedi, “Numerical transient analysis of Markov models,” *Computers and Operations Research*, vol. 15, no. 1, pp. 19–36, 1988.
14. B. Melamed and M. Yadin, “Randomization procedures in the computation of cumulative-time distributions over discrete state Markov processes,” *Operations Research*, vol. 32, pp. 926–944, July–August 1984.
15. J. K. Muppala and K. S. Trivedi, “Numerical transient analysis of finite Markovian queueing systems,” in *Queueing and Related Models* (U. N. Bhat and I. V. Basawa, eds.), pp. 262–284, Oxford University Press, 1992.

A PEPA Workbench model of the Active Badge System

We show the version of the `.pepa` file generated for the active badge model for 3 people and 6 rooms, as passed to the `ipc` tool. Due to space limitations the equivalent 3 people/6 rooms version of the HYDRA model is not shown here but can be found at:

`http://www.doc.ic.ac.uk/ipc/examples/`

In the equivalent `.mod` file, we have expressed a passage specification which represents any of the people ending up in room 6, having started from room 1 (as in the second passage specification in Section 5.2).

```
% Model created by: mkActive - parameters: 3 people, 6 rooms
# Person1 = (reg1, r).Person1 + (move2, m).Person2;
# Person2 = (move1, m).Person1 + (reg2, r).Person2 + (move3, m).Person3;
# Person3 = (move2, m).Person2 + (reg3, r).Person3 + (move4, m).Person4;
# Person4 = (move3, m).Person3 + (reg4, r).Person4 + (move5, m).Person5;
# Person5 = (move4, m).Person4 + (reg5, r).Person5 + (move6, m).Person6;
# Person6 = (reg6, r).Person6 + (move5, m).Person5;
# Sensor1 = (reg1, infty).(rep1, s).Sensor1;
# Sensor2 = (reg2, infty).(rep2, s).Sensor2;
# Sensor3 = (reg3, infty).(rep3, s).Sensor3;
# Sensor4 = (reg4, infty).(rep4, s).Sensor4;
# Sensor5 = (reg5, infty).(rep5, s).Sensor5;
# Sensor6 = (reg6, infty).(rep6, s).Sensor6;
# Dbase1 = (rep1, infty).Dbase1 + (rep2, infty).Dbase2
         + (rep3, infty).Dbase3 + (rep4, infty).Dbase4
         + (rep5, infty).Dbase5 + (rep6, infty).Dbase6;
# Dbase2 = (rep1, infty).Dbase1 + (rep2, infty).Dbase2
```

```
         + (rep3, infty).Dbase3 + (rep4, infty).Dbase4
         + (rep5, infty).Dbase5 + (rep6, infty).Dbase6;
# Dbase3 = (rep1, infty).Dbase1 + (rep2, infty).Dbase2
         + (rep3, infty).Dbase3 + (rep4, infty).Dbase4
         + (rep5, infty).Dbase5 + (rep6, infty).Dbase6;
# Dbase4 = (rep1, infty).Dbase1 + (rep2, infty).Dbase2
         + (rep3, infty).Dbase3 + (rep4, infty).Dbase4
         + (rep5, infty).Dbase5 + (rep6, infty).Dbase6;
# Dbase5 = (rep1, infty).Dbase1 + (rep2, infty).Dbase2
         + (rep3, infty).Dbase3 + (rep4, infty).Dbase4
         + (rep5, infty).Dbase5 + (rep6, infty).Dbase6;
# Dbase6 = (rep1, infty).Dbase1 + (rep2, infty).Dbase2
         + (rep3, infty).Dbase3 + (rep4, infty).Dbase4
         + (rep5, infty).Dbase5 + (rep6, infty).Dbase6;
# Sys = (((Person1 <> (Person1 <> Person1))
        <reg1, reg2, reg3, reg4, reg5, reg6>
        (Sensor1 <> (Sensor2 <> (Sensor3
          <> (Sensor4 <> (Sensor5 <> Sensor6))))))
          <rep1, rep2, rep3, rep4, rep5, rep6> Dbase1);

Sys
```