SeQuaiA: A Scalable Tool for Semi-Quantitative Analysis of Chemical Reaction Networks

Milan Češka[1], Calvin Chau[2], and Jan Křetínský[2]

[1] Brno University of Technology, Brno, Czech Republic
[2] Technical University of Munich, Munich, Germany

Abstract. Chemical reaction networks (CRNs) play a fundamental role in analysis and design of biochemical systems. They induce continuous-time stochastic systems, whose analysis is a computationally intensive task. We present a tool that implements the recently proposed semi-quantitative analysis of CRN. Compared to the proposed theory, the tool implements the analysis so that it is more flexible and more precise. Further, the GUI of the tool offers a wide range of visualization procedures that facilitate the interpretation of the analysis results as well as guidance to refine the analysis. Finally, based on the analysis, we define and implement a new notion of "mean" simulations, summarizing the typical behaviours of the system in a way directly comparable to standard simulations produced by other tools.

1 Introduction

Chemical Reaction Networks (CRNs) are a language widely used for *modelling and analysis* of biochemical systems [10] as well as for high-level programming of molecular devices [33, 6]. They provide a compact formalism equivalent to Petri nets [30], vector addition systems [24] and distributed population protocols [3]. A CRN consists of a set of chemical reactions of given species, each running at a certain rate (intuitively, speed).

Example 1 (Gene expression). Our running example is the classic simple expression of a protein given by the reactions of production (p) and degradation (d) of proteins and blocking (b) the DNA, over three species: protein (P), active DNA (DNA_{on}), and blocked DNA (DNA_{off}):

$$\text{p: } D_{on} \xrightarrow{10} D_{on} + P \qquad \text{d: } P \xrightarrow{0.1} \emptyset \qquad \text{b: } D_{on} + P \xrightarrow{0.001} D_{off}$$

Using mass-action kinetics (the higher the population of reactants, the fastrer the reaction), the CRN induces a infinite population Markov chain in Fig. 1.

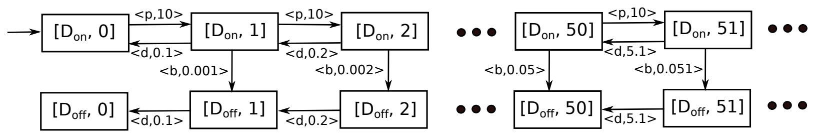

Fig. 1. The Markov chain for Gene expression, displaying the population of P. To simplify the exposition, D_{on} and D_{off} are displayed as discrete "states" of the system, but in fact are two species and the two "states" are just shorthands for 1,0 and 0,1, respectively.

In order to facilitate numerous applications in systems and synthetic biology, various techniques for simulation and formal analysis of CRNs have been proposed, e.g. [15, 32, 18, 7, 2]. We pinpoint several specifics of this setting, necessary to motivate and understand the features of the tool:

1. The analysis is notoriously difficult and **computationally expensive** due to several aspects: *state-space explosion* (exponential growth in the number of species, possibly infinite spaces due to unbounded populations as in Fig. 1, different rates for different populations, again as in Fig. 1), *stochasticity* (races between reactions), *stiffness* (rates of different magnitudes), *multimodality* (qualitatively different behaviours such as extinction of predators only, or also of preys in the predator-prey models) [34, 17]. Consequently, even for small CRNs, simulations may take minutes and analyses hours.
2. We have to face **imprecise inputs**. In particular, even if all relevant reactions are known, the rates are typically not. It is then not clear what behaviours can be induced by all possible values.
3. The analysis **output need not be precise** numerically, but only qualitatively. For instance, it is important to know that initial growth is followed by extinction and what the order of magnitude of the peak population is, but not necessarily what the exact distribution at an exact time is. Unfortunately, it is hard to compute the qualitative information without the quantitative one.
4. Biologists and engineers often seek for plausible **explanations** why the system under study features or not the discussed behaviour. In many cases, a set of system simulations/trajectories or population distributions is not sufficient and the ability to provide an accurate explanation for the temporal or steady-state behaviour is another major challenge for the existing techniques.

SeQuaiA is a tool for analysis of CRN addressing these issues:

1. It features unprecedented **scalability**, analysing standard complex benchmarks within a fraction of a second.
2. It is **robust** w.r.t. concrete rates, not depending on the exact values but only on their orders of magnitude, yet revealing rare behaviours to the desired extent.
3. Its *semi-quantitative analysis* is **precise enough** to conclude on the qualitative behaviour of the system and on rough estimates of the quantities (population sizes, times).
4. It produces models (Markov chains) that are explicit, yet **interpretable**, making the behaviour more **explainable**.

It is based on the technique presented in [9], relying on two cornerstones. Firstly, it computes a system abstraction with **acceleration**, abstracting not only states and single transitions, but taking into account *segments* of paths. The resulting models are small enough to allow for a synoptic observation of the model dynamics. Secondly, it performs **semi-quantitative analysis**, focusing on the most probable behaviours and more qualitative, global descriptions, such as oscillation, rather than fully quantitative sequences of exact transient distributions. This yields explainable models and is a sufficient and computationally cheaper technique. While the basic theory is derived from [9], there are a number of new features and differences in our tool, not just the implementation:

Method: (i) The abstraction is *more precise* now that the tool can also compute numerical outputs, whereas [9] focuses on a manually feasible, and hence imprecise, abstraction. (ii) It suggests how to *refine the abstractions*, providing a knob for trading precision for computational resources.

Visualization: The GUI provides a number of ways to display the results, facilitating understanding the models, including (i) identification of strongly connected parts of 'iterations', corresponding to 'temporarily stable' behaviours, (ii) quantitative information on transient times and steady-state distributions, or (iii) visual qualitative explanations, such as semantic grouping of states or tracking correlations between populations.

Additional analysis instruments: (i) The new notion of *envelope* provides an explicit knob to consider not only the most probable, but also less probable behaviours. (ii) The novel concept of *mean simulation* yields summaries of most probable runs and an analysis output directly comparable to classic simulation-based tools.

Related Work. Since a direct analysis of the Markov chains induced by CRN does not scale well [19], deterministic approximations through fluid (mean-field) techniques can be applied [4, 8] to large populations, but cannot adequately capture the stochasticity of CRNs caused by low population species. To this end, both can be combined in hybrid approaches [21, 18, 7], typically involving a computationally demanding numerical analysis. Reduction techniques such as [12, 1] are based on approximate bisimulation [11], on aggregation acording to the CRN-specific structure [27, 35, 13], or state truncation [29, 20, 28].

Despite the plethora of techniques, the practical analysis often relies on the stochastic simulation [15] and its multi-scale improvements [17, 31, 32, 5, 14, 22]. In contrast, our approach (i) provides a compact explanation of the system behaviour in the form of tiny models allowing for a synoptic observation and (ii) can easily reveal less probable behaviours. The widely used tool for analysis of CRNs are almost exclusively based on stochastic simulations, for instance, the platform-independent Copasi [23], DSD [25] with a convenient web-based graphical interface, or StochPy [26] easily extensible using Python scientific libraries.

2 Workflow and key functionality

In this section, we guide the reader through the workflow, discuss the key features of the tool and demonstrate them on examples. A technical user manual can be found in Appendix. The tool features five tabs, which also reflect the workflow. First, a CRN is either retrieved from a file in the Load model tab or a new one is created. Either way, the model can be changed in the Editor tab together with the analysis parameters. The process continues in the Analysis tab. The analysis follows in two steps. First, the *semi-quantitative abstraction* of the Markov chain for the CRN is generated; second, the *semi-quantitative* analysis is performed on the abstraction. The tool offers an explicit option to display the abstraction as .dot file or to directly run both steps. After the complete analysis is executed, the Visualization tab offers a range of options to *display* the results, including various *quantitative properties*. Finally, the analysed model can be used to generate concrete runs on the Simulation tab, which we call *mean simulations* since they display the "average-case" behaviour. In the following we detail on these key elements.

2.1 Semi-quantitative abstraction

Key idea. The abstraction of the state space is simply given by a discretization of the population for each species into finitely many intervals, see Fig. 2. The classic may abstraction of the transition function results in non-deterministic self-loops as in Fig. 2 (top) in red, which make impossible to conclude anything useful (except for some safety properties) on the behaviour once we reach such a state, even whether it is ever left at all. Instead, [9] considers sequences of transitions: in this case, sequences of prevalently growing transitions (those

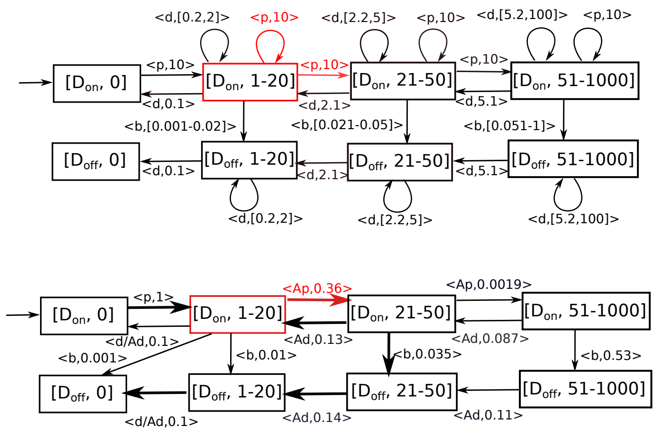

Fig. 2. The abstract Markov chain for Gene expression with population discretization thresholds $20, 50$ with classic may transition function and the semi-quantitative version with accelerated transitions.

increasing the population) are significantly more probable then the prevalently decreasing ones. Consequently, the self-looping transitions are *accelerated* (taken multiple times) to get a "combined" transition that brings a typical representative of this population interval into a higher interval, see Fig. 2 (bottom) also in red. Hence the new rate reflects (i) the mass-action kinetics with the typical population in the interval and (ii) the typical number of the transition repetitions before another interval is reached. These accelerated transitions are the key idea of the semi-quantitative abstraction and are denoted by the tool by a prefix A.

Tool inputs. Technically, the tool requires, for each species, a (possible empty) list of increasing population thresholds $t_1, t_2, \ldots t_n$ and a population bound t_b. The thresholds split the concrete population to the intervals $[0, 0], (0, t_1], (t_1, t_2], \ldots, (t_{n-1}, t_n], (t_n, \infty)$. Here 0 is taken separately to reflect enabledness of actions; the representatives, used for consequent computations, are chosen to be in the middle of the intervals and derived from t_b for the last one. (For the empty list we have only one non-zero interval $(0, \infty)$). The input numbers are supposed to reflect the monitored property of interest and the required precision, the bound t_b should give a probable upper bound on the maximal population. How to obtain and iteratively improve these is discussed in Section 2.5 on refinement.

Example 2. Consider Gene expression, now with a 'fast' blocking where the rate of b equals 10^{-2}. A typical simulation can be seen in Fig. 3 (left): the number of proteins grows until several dozen, then blocking takes place until extinction. The semi-quantitative abstraction for thresholds $10, 20, 50$ yields the model in Fig. 4 (a). In contrast to classic abstractions, there are no self-loops and the abstract transitions are assigned concrete rates. One can see that the blocking can in principle take place at any population and that population can decrease also when DNA is on, i.e. in states $[1, 0, \cdot]$. However, all this happens with very low probabilities and the model captures this only indirectly through the numerical labelling. This is then made explicit during the semi-quantitative analysis.

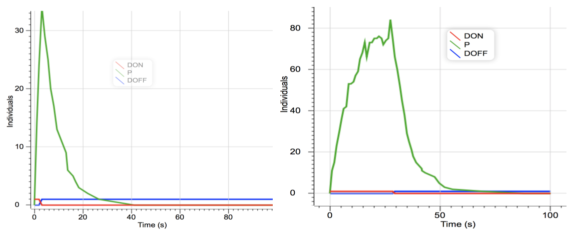

Fig. 3. Simulations of the Gene expression model, obtained using DSD tool [25]. **Left:** Fast variant with rate of b being 10^{-2} **Right:** Slow variant with rate of b being 10^{-3}.

2.2 Semi-quantitative analysis

Key idea. The aim is to prune the abstraction so that only reasonably probable behaviour is reflected, see the thick transitions in the abstraction in Fig. 2 (bottom). To this end, we preserve in each state only the transitions with the highest rate h or almost highest rates, i.e. with $h' > h/envelope$ where $envelope > 1$ is a parameter. Parameter values in $[1, 10]$ ensure we can only look at rates of the same order of magnitude, thus the most probable events and those with e.g. only 20% chance of happening. Higher values then allow for inspection of even less probable behaviours.

Consequently, the method can naturally handle uncertainty in the reaction rates since typically only the relative magnitudes of the rates are important, actually, only their orders of magnitude. This robustness w.r.t. the input is very beneficial for biologists as the precise rates are often not known.

Example 3. The analysis of the previous 'fast' Gene expression with $envelope = 3$ is depicted in Fig. 4 (b). As such it shows the most probable behaviours: the fast growth until the intervals 2 and 3 (i.e. 10-20 and 20-50) and not beyond to 4 (over 50), followed by a slower decline. The computed rates induce expected times to pass through a state, matching closely those of the simulation Fig. 3 (left). Moreover, we see that the blocking transition from interval 2 has a lower probability than the production, is thus less probable. As such it would not even appear as a probable one, for a stricter $envelope = 2$.

Example 4. A more complicated behaviour arises when the blocking is slow, with rate 10^{-3} as in Introduction. A simulation run for this case is depicted in Fig. 3 (right). One can observe a more balanced competition between blocking and oscillation around 70-100 proteins. Similarly, while the abstraction features arbitrary oscillations (also back to no proteins at all), see Fig. 4 (c), after analysis the pruned abstraction is faithfully modelling the initial growth, subsequent oscillation only in the range of higher populations, followed by blocking and gradual extinction of proteins, see Fig. 4 (d).

Technically, the analysis relies on repeated alternation of transient and steady-state analysis. First, starting from the initial state, we follow in each state only the transitions with highest rates (most probable ones), until the set of explored state reaches a fixpoint. A part of the created graph is recurrent and forms a bottom strongly connected component (BSCC) or a collection thereof. The system temporarily settles in the steady state of this BSCC. After some time has passed, also a less probable transition happens almost surely and the "BSCC"

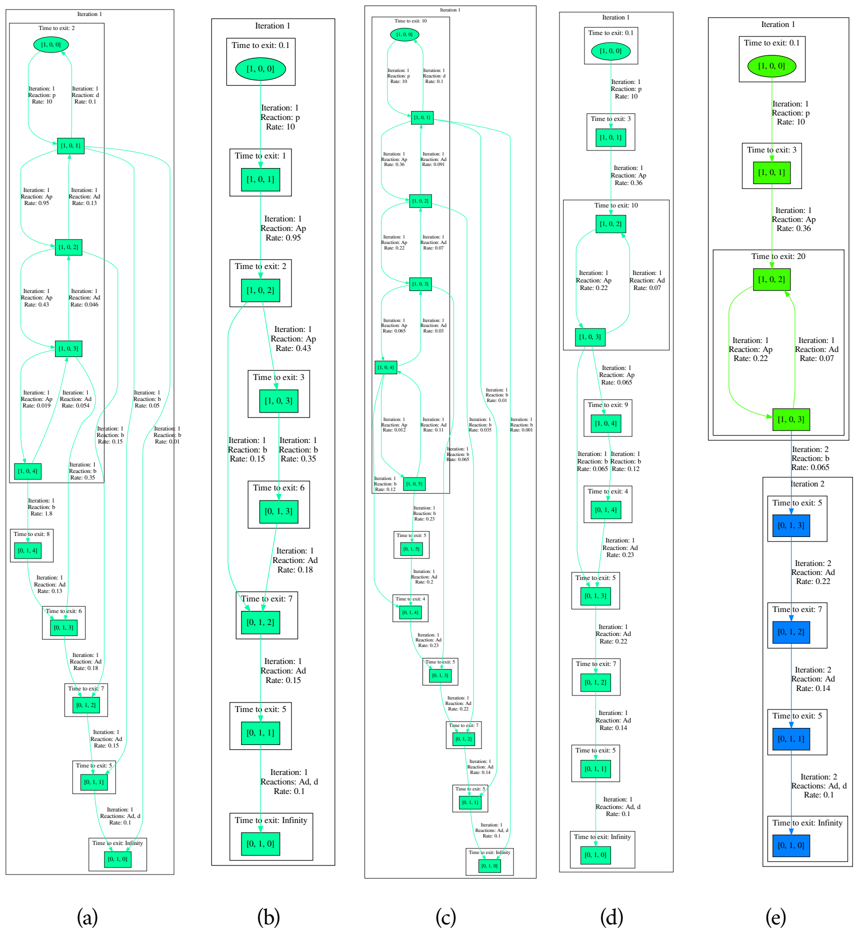

Fig. 4. (a) and (b): 'Fast' Gene expression with thresholds 10, 20, 50. (a) depicts the full abstraction and (b) depicts *envelope* = 3. (c) – (e): 'Slow' Gene expression with thresholds 20, 50, 80, 150. (c) depicts the full abstraction. (d) and (e) depicts *envelope* = 3 and 1, respectively.

is exited. These exit points are identified by a steady-state analysis of the BSCC, taking the magnitudes of exiting and non-exiting transition rates into account. The exit points trigger a new *iteration* of the transient and then the steady-state analysis.

Example 5. Fig. 4 (e) illustrates a situation with two iteration using the the slow variant of the model. Decreasing *envelope* to 1 caused that the blocking reaction is explored in the second iteration – as an exit of the BSCC found in the first iteration. Before that exit happens, the "BSCC" represents a "temporary" steady state of the system.

2.3 Visualization of qualitative information

A proper visualization is essential for clear presentation and easy interpretation of the results of our analysis. To this end, the tool and its GUI offer various options for visualizing the results. The basic ones, related to the graph structure, are the following. Further options, with more quantitative flavour, are discussed in the next section, followed by an example illustrating all of them.

Iterations. As the complete abstract model is typically very large and chaotic, further structuring is necessary. Therefore, the default view shows the states arranged and grouped into separate blocks, one for each iteration, additionally coloured distinctly for each iteration. Besides, we can restrict which iterations we show. This is useful to zoom in and investigate a particular part of the behaviour.

Intra-iteration SCCs (IISCCs). Additionally, the arrangement and colouring can be based on aggregating SCCs *within* each iteration (IISCCs). This helps to understand the emergence of repetitive behaviour patterns, such as oscillation, steady state or temporary steady state. It can be also be combined with the iteration grouping.

Collapsed views. In order to understand the system behaviour, one typically needs to have a synoptic overview of the system. For more complex systems, even the pruned abstraction could become too large and the view of the fully expanded system might not be sufficiently compact. In such cases, the aggregates discussed in the previous views, i.e., iterations and IISCCs, can be collapsed into a single nodes, hiding the complexity of the exact behaviour pattern within these areas. This allows us, for instance, to ignore the particular (temporary) oscillation or steady state in these states and to focus on more global behaviour, such as what happened before and after this behaviour and how often does it arise. In contrast to zooming in by restricting to certain iteration(s) only, the collapsed views provide a means to zoom out.

2.4 Visualization of quantitative information

The produced graphs are also labelled by *numerical information*. While the quantities cannot be precise due to the simplifications of the extremely scalable analysis, they match the orders of magnitudes of the observed quantities, which is often precise enough for biological purposes; for instance, the peak of protein growth happens after units vs. dozens of seconds in the fast and slow variants of Gene expression, respectively.

Transient analysis. Firstly, each abstract transition is labelled with a rate corresponding (in the order of magnitude) to the rate of the concrete transition (or accelerated transition, i.e. a "sequence" of transitions) of a "typical" representative of the abstract state. These rates induce the expected time spent in each transient state of each iteration. Indeed, the waiting time is simply the inverse of the sum of the outgoing rates. Further, each BSCC of each iteration is labelled by an estimate of time before it is left into the next iteration. This is a key notion, which allows us to easily provide transient timing information for very stiff systems (working at different time scales). Consider the simple gene model. From Fig. 4 (b) and (d) we can easily compute the expected time to the extinction (as the sum of the exit time for all SCC on the inspected path). Our analysis correctly estimates that the expected extinction time is around 24 and for the fast variant and 40 for the slow variant.

Steady state analysis. In many biological models, the natural steady state is either extinction or unbounded explosion. Hence it does not say much about the "seemingly steady" state (the temporary steady state), i.e., behaviour that is stable for a long but finite time. For that reason,

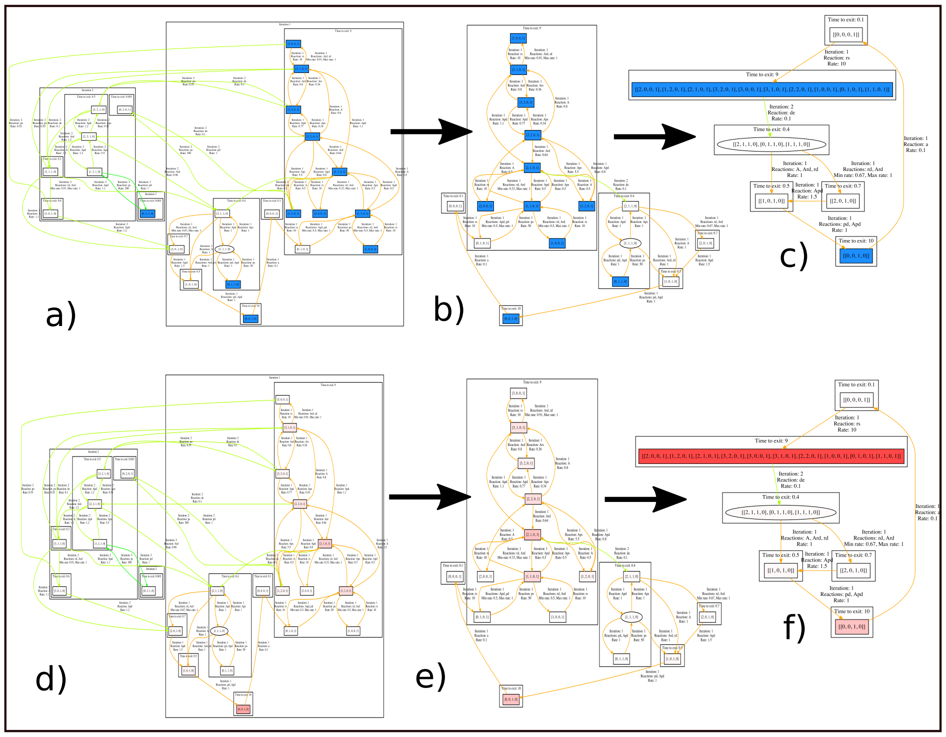

Fig. 5. Extended gene expression – abstraction, iteration, steady state, correlation

the tool provides information not only on the steady state of the whole system, but also for each iteration separately since they represent the temporary steady states discussed above. Both can be conveniently visualized as colouring of states, with higher probabilities corresponding to darker colours, immediatelly giving a synoptic view on frequent behaviours.

Correlations. Finally, correlations between population sizes can be observed as follows. The GUI can be given a set of equivalences of the form $m \sim n$ for species i, j, meaning that if a state has (abstract) population m of species i and n of j then it is regarded as satisfying the correlation in question. It is coloured accordingly and the overall colouring of the system provides further indication under which behaviour or in which phases the correlation holds.

Example 6. We demonstrate these visualization options on a more complicated gene expression model [16, 18], widely used model for benchmarking CRN analyzers, in Fig. 5. The behaviour oscillates between two steady states with DNA on and DNA off. Moreover, there is a correlation between high amounts of RNA present and DNA being on, and no RNA with DNA off.

The complete system is depicted in the left column, using the iteration and IISCC arrangement. The lower pictures always depict the steady state, the upper ones the correlation. Hence d) shows immediately without seeing any details that the only interesting states are in iteration 1 (the big box) and a) shows in blue where the correlation holds: almost exclusively in this iteration. Therefore, we can zoom in to iteration 1 only (middle column), this time with IISCC arrangement and again depicting both correlation and steady state. Now it is even clearer that most of the time we spend in the correlated states. Finally, in the right column, while zoomed in to iteration 1, we zoom out by collapsing the IISCCs, ignoring the internal (non-interesting) behaviour of the big IISCC and only observing the interesting switches between the two temporary steady states.

2.5 Precision and refinement

So far, we have illustrated the concepts and the functionality on models with an appropriate level of abstraction. However, it often happens that we start the investigations with a too coarse abstraction. Whenever this happens, it is important to notice this and appropriately refine the abstraction. While [9] does not discuss this issue, the tool provides support also for that.

Precision parameters There are several knobs for trading the size and the precision of the abstraction. They all come as input in the lower half of the Editor tab: discretization, bound, and envelope.

Example 7. Recall the initial abstraction for the Gene expression of Fig. 2 (with rate 10^{-3}). The abstraction, using thresholds $20, 50$ predicts an oscillation including low populations of P (1-20) which is not correct (recall that the P oscillates on high populations before the blocking reaction occurs). Fig. 4 (c) and (d) show the abstraction and the consequent analysis and visualization for a refined model using thresholds $20, 50, 80, 150$ (instead of just $20, 50$). As already discussed, this abstraction already correctly predicts the system behaviour.

Discretization The basic building block of each abstraction is the degree of details it preserves in the abstract states. Firstly, it determines how precisely we can observe the evolution of the population. For instance, whenever we want to detect whether a population typically grows beyond a bound or oscillates in a certain interval, such an interval should be present in the discretization. Secondly, the discretization should be fine enough so that in each state, the rates are reasonably (in orders of magnitude) precise. Fortunately, in our analysis their absolute precision is not vital. In contrast, we only need *relative* proportions of the rates to have the right *magnitude* to decide which behaviour is probable. Consequently, too rough abstraction is reflected in *"non-determinism"* when a state has two transitions under similar rate. In such a case, the probable behaviour cannot be determined. Therefore, the Visualization tab provides in the Colorization pane an option to provide suggestions for refinement, including highlighting non-deterministic states, pointing at the natural candidates for refinement. Note that we highlight only the states where the two transitions lead to mutually different SCCs so that a significant change in behaviour may occur.

Bounds Similarly, for the single infinite interval (t_n, ∞), the tool inputs a *bound* which is a believed safe upper bound on the population of the species. Of course, it may be wrong. This is irrelevant in case when the population explodes beyond all bounds. However, whenever there are transitions from the highest level back to a lower one, its feasibility and rate are in question. Optimally, such states do not even occur in the pruned abstraction. If they do, we also highlight them using the Colorization for Refinement suggestions (in another colour).

Envelope As too rough abstractions introduce too much non-determinism, dually, the degree of the non-determinism is determined (even defined) by the *envelope*, the factor between rates so that even the less probable option is still taken into account (and thus introduces non-determinism). Consequently, high values of envelope introduce non-determinism, making the analysis take also less important behaviour into account; in contrast, low values make the analyzed system deterministic, showing only the most probable behaviour. The choice of the envelope thus depends on whether such behaviours should also be reported. Using several values, we can then easily determine which behaviours are rarer than others.

2.6 Mean simulations

Since our models, although abstract, have an operational semantics, we can even run simulations on them. Moreover, the accelerated transitions, as "sequences" of transitions, have a low variance in the expected time, by the law of large numbers. Hence their execution time can be chosen quite precisely in a deterministic way. Similarly, the time to leave an IIBSCC is quite deterministic. Thus we can generate simulation where the only random decisions are choices of transitions, but the timing follows the mean time of the respective events. Moreover, runs within the pruned abstraction reflect the most important behaviours only. Altogether, such *mean simulations*[3], which can thus be generated from our analysis, represent groups of typical runs (modulo small time shifts and order of transitions within an SCC, which are not very relevant). Therefore, a few such simulation reflect all the present behaviours (on a level of desired significant probability) and can serve to observe multi-modalities, bifurcations, rough transient timing as well as frequencies in the steady-state and temporary steady-state.

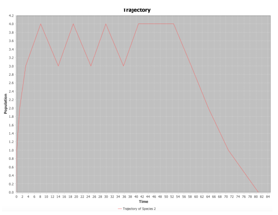

Fig. 6. Mean simulation for the slow variant of Gene expression, directly comparable to Fig. 3 (right).

Example 8. Fig. 6 shows an abstract simulation for our running example. One can readily observe its validity with respect to the typical stochastic simulation in Fig. 3 (right).

3 Conclusion

We have presented a scalable tool for robust and explainable analysis of CRN. The analysis is precise enough as cross-validated with simulation-based results on several models widely used in the literature. The analysis of each model took less than a second. One of the key contributions of the tool is the visualization, which is essential for clear presentation and easy interpretation of the results of our analysis.

While the precision is experimentally good, theoretical error bounds are an interesting future direction, also usable for automating the refinement. Finally, the techniques could be helpful in the synthesis of CRN.

[3] They are not means of values from simulations since averaging oscillating values may result in no oscillation. Rather they reflect "mean patterns".

References

1. Abate, A., Katoen, J.P., Lygeros, J., Prandini, M.: Approximate model checking of stochastic hybrid systems. European Journal of Control **16**, 624–641 (2010)
2. Abate, A., Brim, L., Češka, M., Kwiatkowska, M.: Adaptive aggregation of markov chains: Quantitative analysis of chemical reaction networks. In: Computer Aided Verification (CAV), pp. 195–213. Springer (2015)
3. Angluin, D., Aspnes, J., Eisenstat, D., Ruppert, E.: The computational power of population protocols. Distributed Computing **20**(4), 279–304 (2007)
4. Bortolussi, L., Hillston, J.: Fluid model checking. In: Concurrency Theory (CONCUR), pp. 333–347. Springer (2012)
5. Cao, Y., Gillespie, D.T., Petzold, L.R.: The slow-scale stochastic simulation algorithm. The Journal of chemical physics **122**(1), 014116 (2005)
6. Cardelli, L.: Two-domain DNA strand displacement. Mathematical Structures in Computer Science **23**(02), 247–271 (2013)
7. Cardelli, L., Kwiatkowska, M., Laurenti, L.: A stochastic hybrid approximation for chemical kinetics based on the linear noise approximation. In: Computational Methods in Systems Biology (CMSB). pp. 147–167. Springer (2016)
8. Cardelli, L., Tribastone, M., Tschaikowski, M., Vandin, A.: Maximal aggregation of polynomial dynamical systems. Proceedings of the National Academy of Sciences **114**(38), 10029–10034 (2017)
9. Češka, M., Křetínský, J.: Semi-quantitative abstraction and analysis of chemical reaction networks. In: CAV'19. Springer International Publishing (2019)
10. Chellaboina, V., Bhat, S.P., Haddad, W.M., Bernstein, D.S.: Modeling and analysis of mass-action kinetics. IEEE Control Systems Magazine **29**(4), 60–78 (2009)
11. Desharnais, J., Laviolette, F., Tracol, M.: Approximate analysis of probabilistic processes: logic, simulation and games. In: Quantitative Evaluation of SysTems (QEST). pp. 264–273. IEEE (2008)
12. D'Innocenzo, A., Abate, A., Katoen, J.P.: Robust PCTL model checking. In: Hybrid Systems: Computation and Control (HSCC). pp. 275–285. ACM (2012)
13. Ferm, L., Lötstedt, P.: Adaptive solution of the master equation in low dimensions. Applied Numerical Mathematics **59**(1), 187–204 (2009)
14. Ganguly, A., Altintan, D., Koeppl, H.: Jump-diffusion approximation of stochastic reaction dynamics: error bounds and algorithms. Multiscale Modeling & Simulation **13**(4), 1390–1419 (2015)
15. Gillespie, D.T.: Exact stochastic simulation of coupled chemical reactions. The journal of physical chemistry **81**(25), 2340–2361 (1977)
16. Golding, I., Paulsson, J., Zawilski, S.M., Cox, E.C.: Real-time kinetics of gene activity in individual bacteria. Cell **123**(6), 1025–1036 (2005)
17. Goutsias, J.: Quasiequilibrium approximation of fast reaction kinetics in stochastic biochemical systems. The Journal of chemical physics **122**(18), 184102 (2005)
18. Hasenauer, J., Wolf, V., Kazeroonian, A., Theis, F.: Method of conditional moments (MCM) for the chemical master equation. Journal of Mathematical Biology pp. 1–49 (2013). https://doi.org/10.1007/s00285-013-0711-5
19. Heath, J., Kwiatkowska, M., Norman, G., Parker, D., Tymchyshyn, O.: Probabilistic model checking of complex biological pathways. Theoretical Computer Science **391**(3), 239–257 (2008)
20. Henzinger, T.A., Mateescu, M., Wolf, V.: Sliding Window Abstraction for Infinite Markov Chains. In: Computer Aided Verification (CAV), pp. 337–352. Springer (2009)
21. Henzinger, T.A., Mikeev, L., Mateescu, M., Wolf, V.: Hybrid numerical solution of the chemical master equation. In: Computational Methods in Systems Biology (CMSB). pp. 55–65. ACM (2010)
22. Hepp, B., Gupta, A., Khammash, M.: Adaptive hybrid simulations for multiscale stochastic reaction networks. The Journal of chemical physics **142**(3), 034118 (2015)

23. Hoops, Stefan and Sahle, Sven and Gauges, Ralph and Lee, Christine and Pahle, Jürgen and Simus, Natalia and Singhal, Mudita and Xu, Liang and Mendes, Pedro and Kummer, Ursula: COPASI - a COmplex PAthway SImulator. Bioinformatics **22**(24), 3067–3074 (2006). https://doi.org/10.1093/bioinformatics/btl485, http://bioinformatics.oxfordjournals.org/content/22/24/3067.abstract
24. Karp, R.M., Miller, R.E.: Parallel program schemata. Journal of Computer and system Sciences **3**(2), 147–195 (1969)
25. Lakin, M.R., Youssef, S., Polo, F., Emmott, S., Phillips, A.: Visual DSD: a design and analysis tool for dna strand displacement systems. Bioinformatics **27**(22), 3211–3213 (2011)
26. Maarleveld, T.R., Olivier, B.G., Bruggeman, F.J.: StochPy: a comprehensive, user-friendly tool for simulating stochastic biological processes. PloS one **8**(11) (2013)
27. Madsen, C., Myers, C., Roehner, N., Winstead, C., Zhang, Z.: Utilizing stochastic model checking to analyze genetic circuits. In: Computational Intelligence in Bioinformatics and Computational Biology (CIBCB). pp. 379–386. IEEE (2012)
28. Mateescu, M., Wolf, V., Didier, F., Henzinger, T.A.: Fast Adaptive Uniformization of the Chemical Master Equation. IET Systems Biology **4**(6), 441–452 (2010)
29. Munsky, B., Khammash, M.: The finite state projection algorithm for the solution of the chemical master equation. The Journal of chemical physics **124**, 044104 (2006)
30. Murata, T.: Petri nets: Properties, analysis and applications. Proceedings of the IEEE 77(4), 541–580 (1989)
31. Rao, C.V., Arkin, A.P.: Stochastic chemical kinetics and the quasi-steady-state assumption: application to the gillespie algorithm. The Journal of chemical physics **118**(11), 4999–5010 (2003)
32. Salis, H., Kaznessis, Y.: Accurate hybrid stochastic simulation of a system of coupled chemical or biochemical reactions. The Journal of chemical physics **122**(5), 054103 (2005)
33. Soloveichik, D., Seelig, G., Winfree, E.: DNA as a universal substrate for chemical kinetics. Proceedings of the National Academy of Sciences of the United States of America **107**(12), 5393–5398 (2010)
34. Van Kampen, N.G.: Stochastic processes in physics and chemistry, vol. 1. Elsevier (1992)
35. Zhang, J., Watson, L.T., Cao, Y.: Adaptive aggregation method for the chemical master equation. International Journal of Computational Biology and Drug Design **2**(2), 134–148 (2009)

A Quick user manual

- 1) General information
  The tool has been developed with Java 11 and is provided as an executable jar. In the following we will walk you through the tool and make you familiar with its features.
- 2) Model tab
  In the model tab you can load a model from a file. To load a model:
  1) Click on open file, the a file chooser dialog opens
  2) Select a file you want to open and click open to confirm your selection
  3) The information about the model is displayed below, i.e. name of the model, number of species and the number of reactions.
  You can now proceed to the next steps, by choosing one of the other tabs.
- 3) Editor tab
  If you would like to edit a model which is stored in a file, please go to the model tab and proceed as described above. The editor offers the possibility to set the name of your model, the number of species, the initial states and and allows you to add/edit/remove the reactions and bounds and population discretisation.
- 3.1) Editing the number of species
  Please note that editing the number of species with the spinner invalidates the reactions, initial state, bounds and discretisation. Thus, everything is reset once you change the number of species.
- 3.2) Adding reactions
  To add a reaction:
  1) Click on the "Add"-button
  2) A dialog opens where you can edit the reaction
  3) Next to "Reaction label:", you can edit the label for the reaction
  4) Next to "Reactant(s):", you can edit the coefficients for the reactants. The first input field corresponds to the coefficient for the first species, the second corresponds to the second species and so forth. The coefficient for reactants should be given as non-positive integers.
  5) Next to "Product(s):", you can edit the coefficients for the products. The first input field corresponds to the coefficient for the first species, the second corresponds to the second species and so forth. The coefficient for reactants should be given as non-negative integers.
  6) Next to "Rate:", you can edit the rate, which is given as a floating number.
  7) Click the "Add reaction"-button to add the reaction
- 3.3) Editing the initial state
  1) Click "Edit" in the "Init state" section.
  2) A dialog opens, the first input field corresponds to the first species, the second input field corresponds to the second species and so forth. Please enter a concrete population value for each field, e.g. if you would like to start with 1000 individuals of the first species, you would simply enter 1000 in the first input field.
  3) Click the "Apply"-button to apply the changes. You can close the window afterwards.
- 3.4) Editing the bounds
  1) Click the "Edit bounds"-button
  2) A dialog opens, the first input field corresponds to the first species, the second input field corresponds to the second species and so forth. Please enter a concrete population

value for each field, e.g. if you would like to bound the population of the first species by 1000 individuals, you would simply enter 1000 in the first input field.
3) Click the "Apply"-button to apply the changes. You can close the window afterwards.

- 3.5) Editing the population discretization
  1) Click the "Edit discretization"-button
  2) A dialog opens, choose a species for which you would like to edit the population levels.
  3) To insert a new population level, simply enter the concrete population value into the input field and click the "Add"-button. The level is sorted accordingly.
  Interpretation of the list of values: An empty list corresponds to the population levels 0 and 1. Suppose a list consists of the values 0,10 and 20, then we would have 4 population levels 0,1-10,11-20 and 21-boundValue, where boundValue is edited with the "Edit bounds"-button (see above).
- 3.6) Editing the envelope
  Simply edit the input field in the envelope section. You can enter a non-negative floating point number.
- 3.7) Saving the model
  1) Click the "Choose file"-button
  2) A file chooser dialog opens, select a file and confirm your selection with "Open".
  3) Click the "Save"-button
- 4) Analysis tab
  Note that before you can only proceed with any of the actions below, if you have loaded a model. Further, note that everytime you load a new model you will need to perform the analysis again.
- 4.1) Saving the abstraction
  1) Click the "Choose file"-button
  2) A file chooser dialog opens, select a file and confirm your selection with "Open".
  3) Click the "Generate DOT file for raw abstraction"-button to save
- 4.2) Performing the analysis
  1) Click on the "Perform analysis"-button
  2) The text-area below the button will provide you with feedback
- 5) Visualization tab
  Note that before you can only proceed with any of the actions below, if you have performed an analysis.
- 5.1) Choosing min-iteration and max-iteration
  You can enter integer values for minimum iteration and maximum iteration. The visualization will then only display states discovered within the given iteration range. Click on the "Reset"-button to set minimum iteration to 0 and maximum iteration to 2,147,483,647.
- 5.2) Grouping the states
  Go to the "Grouping" section and choose how you would like to group the states. "Group by iteration and iteration SCCs" means that the states are first grouped according to the iteration they are first explored in. Within the iteration, they are then grouped according to the strongly-connected components they belong to. "Group by iteration" would only group the states according to iterations. "Group by iteration SCCs" would group the states that belong to the same SCC within an iteration together.
- 5.3) Colorization of the states

1) By default "Colorize by iteration" is selected, which colorizes the states according to the iteration they belong to, i.e. states with same color have been first explored within the same iteration.
2) "Colorize steady state" colorizes the steady-steady distribution of the whole system. The intensity of red reflects the probability of ending up in the respective state. The more intensive the color the higher the probability.
3) "Colorize steady state in SCCs" colorizes the steady-state distribution within the SCCs, so called intra-iteration SCCs.
4) "Colorize refinement suggestions" colorizes states in blue with non-deterministic behavior, i.e. states that have at least two outgoing transitions with similar rates (w.r.t. magnitude) to two different targets. Further it colorizes states in red which show high-population degradation. Meaning, states that have at least one species with highest population level and at least one transition that decreases the population level of this particular species.
5) "Colorize correlation" colorizes given correlated values. If you choose this colorization, two buttons appear below, with which you can add/delete correlated values for two pairs of species. Clicking on the button opens a dialog. On top you can select the species for which the correlated values should hold. Below you can enter the values. E.g. if you have selected "Species 1" and "Species 2" and enter value "1" and "2", all states with population 1 for species 1 and population 2 for species 2 will be colorized.
If states satisfy the first correlation, they are colored in blue, if they satisfy the second in yellow and if both then green.
6) "No colorization" does not colorize any state

– 5.4) Selecting the view
1) "Expanded view" will show individual states and their respective grouping
2) "Collapsed aggregates" will show states belonging to one group in one state together. Transitions within the same group are omitted.
– 5.5) Saving the file
1) Click the "Choose file"-button and select a file in the dialog.
2) Click the "Generate DOT file"-button
– 6) Simulation tab
Note that you can only perform the actions below, if you have performed an analysis.
– 6.1) Running a simulation
1) Set the time bound by changing the input field in the "Time bound" section, by default the value is 100.
2) Click the "Run simulation"-button
– 6.2) Plotting the trajectory/distribution
1) Below "Monitor species", you can select the species you want to monitor.
2) Click the "Plot trajectory"- or "Plot distribution"-button.