Supplementary Material

Specifying ChemTest Oracles in Temporal Logic

The stochastic chemical reaction network model is equivalent in behavior to a continuous-time Markov chain (CTMC). As a result, temporal logic is a natural choice for formally specifying properties in our testing suite. *Linear temporal logic* (*LTL*) is a rich logic that classifies the paths of a Markov chain and is especially useful for our purposes. The structure of an LTL formula ϕ can be defined recursively in the following Backus–Naur form:

$$\phi := \text{true} \mid a \mid \neg\phi \mid \phi_1 \wedge \phi_2 \mid X\ \phi \mid \phi_1\ U\ \phi_2. \tag{1}$$

Linear temporal logic formulas [1] specify constraints on infinite paths $\omega = (s_1, s_2, \ldots)$ through a Markov chain where each s_i is a *state*. In equation (2), a is an *atomic proposition* which evaluates to true if the first state s_1 of the path satisfies the proposition a; $X\ \phi$ says that ϕ is true in the *next* state of ω, i.e., that $(s_2, s_3, \ldots)$ satisfies ϕ; and $\phi_1\ U\ \phi_2$ says that ϕ_2 eventually holds starting at some future state s_i and that ϕ_1 holds for every state $s_1, \ldots, s_{i-1}$. Two commonly used operators are *future* defined by $F\ \phi := \text{true}\ U\ \phi$ and *globally* defined by $G\ \phi := \neg F \neg\phi$. Intuitively, $F\ \phi$ is true if there exists a future state s_i that satisfies ϕ and $G\ \phi$ is true if every state s_i in the path satisfies ϕ.

Although linear temporal logic is a natural foundation for specifying properties for CRNs, it has several inadequacies. First, CRNs are modeled with *continuous time* Markov chains (CTMCs), and the temporal operators of LTL do not natively support properties with real-valued time bounds. Second, CRNs are inherently probabilistic and so a notion of expressing the *frequency* for which a formula ϕ is satisfied over many random paths is necessary. Third, many test cases of interest are metamorphic which requires specifying a relationship between two CTMCs. Finally, performance test cases require comparing the real-valued *time* at which certain events occur.

To overcome these inadequacies, we augment LTL so that formulas have the structure

$$\phi := \text{true} \mid a \mid \neg\phi \mid \phi_1 \wedge \phi_2 \mid X\ \phi \mid \phi_1\ U_{\leq t}\ \phi_2. \tag{2}$$

Now the operator $\phi_1\ U_{\leq t}\ \phi_2$ states that ϕ_2 must become true in at most time $t \in [0, \infty]$. Furthermore, we include two new operators that evaluate to a numeric value. The *frequency* operator $\#\ \phi$ defines a formula over a set of paths and returns the fraction of paths satisfying ϕ; the *when* operator $H\ \phi$ is defined over a single path and calculates the real-valued time that ϕ becomes true for the first time or ∞ if ϕ never holds.

As an example, we use the approximate majority program from our study. This program determines which of two species, $X1$ and $X2$, has the initial majority of the population (on input) by annihilating the minority species. This simple description of the problem is the *informal system requirement* that must be implemented and satisfied. We use the following bimolecular algorithm for computing approximate majority [3, 4].

$$\begin{aligned}
X1 + X2 &\xrightarrow{1} X1 + U \\
X1 + X2 &\xrightarrow{1} U + X2 \\
X1 + U &\xrightarrow{1} 2X1 \\
X2 + U &\xrightarrow{1} 2X2.
\end{aligned}$$

Our formulas are written over the input species (in this case $X1$ and $X2$) of the system. For example, one of our test requirements is to determine whether $X1$ or $X2$ has the initial majority by annihilating the minority species. This is written as:

$$\text{True} \rightarrow FG\ (X1 \text{ wins} \vee X2 \text{ wins}). \tag{3}$$

It states that eventually (future) the following condition will hold forever (globally): either X1 or X2 wins. Here "$X1$ wins" is an atomic proposition that means that "all $X2$'s have been converted to $X1$'s," i.e., that $X2$ was annihilated by $X1$. Most of our test requirements have *constraints* that must be satisfied for the property to hold and are of the form $\phi \rightarrow \psi$, so we include the redundant True $\rightarrow$ in the formula of (3) to be consistent with the other test requirements.

In the first iteration of ChemTest we have manually created these specifications. We leave automated generation from the model as future work. We present additional examples from our case study in Table 1 and discuss them below.

Table 1: Example formulas from our case study

Test #	Abstract Test Cases	Type	Description
1	$X1[0] \geq X2[0] \rightarrow FG\ (Y[t] = X1[0] - X2[0])$	F	If initially the number of $X1$ is at least $X2$, then Y is eventually converges to $X1 - X2$
5	$(X1[0]\ \%\ 2 = 0)\ \wedge\ (X1'[0] = 2 \cdot X1[0] + 1)$ $\rightarrow FG\ (Y'[t] > Y[t])$	M	If the input $X1$ on the first trace is even and the input $X1'$ on the second trace is a larger odd number, then the output Y on the second trace is eventually always larger than the output Y on the first trace.
12	$X1[0] > X2'[0] > X2[0]$ $\rightarrow \#FG\ (X1' \text{ wins}) < \#FG\ (X1 \text{ wins})$	H	$X1$ is fixed. When taken over several runs, if the difference between $X1$ and $X2$ in the first trace is greater than the difference of $X1$ and $X2$ in the second trace, then $X1$ in the first trace eventually always "wins" more often than $X1$ in the second trace.

After we formalize test requirements, we generate abstract test cases. The input/output species are determined by the CRN. A test requirement specifies *constraints* (left side of implication) and the oracle (right side of implication). In the previous example the test requirement has no constraints on inputs (hence it is always true). We define four types of abstract tests. The first are functional tests (F). These use a single set of inputs, which have a known output (stable or probabilistic). Table 1 shows three test requirements, each from one of our study subjects. The test number corresponds to the abstract test number for that subject. Test 1 is a functional test for *subtraction*. It has the constraint that $X1$ is greater than $X2$ at the start of the program. The input species are $X1$ and $X2$. The output species is Y and the oracle is that Y is *future globally* equal to the result of $X1 - X2$. We also use metamorphic tests. Metamorphic tests [2, 5, 6] are used when we don't have a clear functional output or when we want to diversify the test suite. It includes two different input sets, evaluated based on the relationship of the outputs. Test 5 from Table 1 is an example of a metamorphic test from our second subject, *hailstone*. The first test input is even and the second is odd and larger than the first. We expect that the output of the second test input, Y', will *future globally* always be larger than the original output Y.

References

1. Christel Baier and Joost-Pieter Katoen. *Principles of Model Checking (Representation and Mind Series)*. The MIT Press, 2008.

2. Tsong Yueh Chen, Fei-Ching Kuo, Huai Liu, Pak-Lok Poon, Dave Towey, T. H. Tse, and Zhi Quan Zhou. Metamorphic testing: A review of challenges and opportunities. *ACM Comput. Surv.*, 51(1):4:1–4:27, January 2018.

3. Anne Condon, Monir Hajiaghayi, David G. Kirkpatrick, and Ján Manuch. Simplifying analyses of chemical reaction networks for approximate majority. In *Proceedings of the 23rd International Conference on DNA Computing and Molecular Programming*, volume 10467 of *Lecture Notes in Computer Science*, pages 188–209. Springer, 2017.

4. Titus H. Klinge. Robust signal restoration in chemical reaction networks. In *Proceedings of the 3rd International Conference on Nanoscale Computing and Communication*, pages 6:1–6:6. ACM, 2016.

5. Anders Lundgren and Upulee Kanewala. Experiences of testing bioinformatics programs for detecting subtle faults. In *Proceedings of the*

International Workshop on Software Engineering for Science - SE4Science '16, pages 16–22, 2016.

6. S. Segura, G. Fraser, A. B. Sanchez, and A. Ruiz-Cortés. A survey on metamorphic testing. *IEEE TSE*, 42(9):805–824, Sept 2016.