Investigation of a Biological Repair Scheme

Vincent Danos[1], Jérôme Féret[2], Walter Fontana[3],
Russell Harmer[4], and Jean Krivine[3]

[1] University of Edinburgh
[2] INRIA, CNRS, École Normale Supérieure
[3] Harvard Medical School
[4] CNRS, Université Paris-Diderot

Abstract. This note details an interaction pattern for the allocation of a scarce biological resource where and when it is needed. It is entirely based on a mass action stochastic dynamics. Domain-domain binding plays a crucial role in the design of the pattern which we therefore present using a rule-based approach where binding is an explicit primitive. We also a use a series of refinements, starting from a very simple interaction set, which we feel gives an interesting and intuitive rationale for the working of the final repair scheme.

1 Introduction

What is a scheme the reader will ask. As we understand it in this paper, a scheme or a pattern is a set of interactions between biological agents (we use the neutral term agent throughout this paper, but one can think of them as idealised proteins) which has a noteworthy property and which one can study and recognise under various guises in various systems. A well-known example is that of a reversible covalent modification system that can be made to behave, under well-chosen conditions, as an ultra-sensitive switch [1]. Another related one is that of an autophosphorylating kinase that shows bistable behaviour and can be used as a memory [2]. Importantly, the often non-intuitive kinetic and topological aspects of biological schemes are integral to their functioning. As such, they differ from static network motifs à la Alon [3] which are singled out because of their statistical properties, not their dynamical ones.

Consider a scheme where an agent $\mathtt{X}$ that can be activated by two upstream 'input' agents $\mathtt{I}_1$, $\mathtt{I}_2$, and can in turn activate two downstream 'output' ones $\mathtt{O}_1$, $\mathtt{O}_2$. Suppose that in the sole presence of $\mathtt{I}_1$ only $\mathtt{O}_1$ gets significantly activated. From these simple premisses one can obtain a wide range of behaviour. Specifically, $\mathtt{I}_2$ can be made to completely override the effect of $\mathtt{I}_1$, meaning that when $\mathtt{I}_2$ is present the only significant amount of downstream activity is that of $\mathtt{O}_2$ irrespective of the presence of $\mathtt{I}_1$. However this is only when the interaction rates, copy numbers (eg $\mathtt{X}$ must be saturated by the inputs $\mathtt{I}$s), and binding interfaces (eg $\mathtt{X}$ must use a different binding interface for the inputs $\mathtt{I}$s and for the outputs $\mathtt{O}$s) are well chosen. Thus the corresponding static motif would retain very little

D. Corne et al. (Eds.): WMC9 2008, LNCS 5391, pp. 1–12, 2009.

of the information on which the scheme hinges, it would be too abstract a view on it. Indeed many behaviours can be extracted even from the basic feed-forward motif [4] (see also Ref. [5] for an extensive discussion of the under-determination of motifs).

The situation is similar for the scheme we will define in this note. One has a target agent `T` that sporadically decays into a bad state. Repair is provided by an agent `K`, but in some rare cases also needs a helper agent `H`. The question is how to allocate this scarce and scarcely needed repair resource `H` so that repair proceeds efficiently. In a world of centralised computations, to get that resource when and where it is needed is a trivial question, as one can just order the `H`s to go where they are needed. Our scheme offers a distributed computational solution that relies entirely on mass action (ie on chance collisions). Incidentally, this pattern was taken form a larger model where it is combined with the setting up of a transient memory to obtain what one might call and error-correcting mechanism.[1] Indeed schemes will often be combined in a biological situation to obtain interesting effects. We are not addressing here the interesting and difficult question of how schemes should be composed, though.

There comes the question of what notation or language to use to represent and study such schemes. As said, they often use in a crucial fashion the dynamics of domain-domain binding -which means both the kinetic aspects of binding and the topological constraints induced by the agents' domain structures which may or may not allow simultaneous binding. There is a recent and growing recognition that such structural detail matters in the statistics of the static properties of protein networks [6]. We believe the same broadly holds for the understanding of their dynamics and have developed in the Kappa project [7] a simple notation for binding which is particularly useful for combinatorial situations (as eg in cellular signalling) and which we will use here. Note that Kappa is one of the few languages to propose such a direct notation (see also the BNG language [8]). Basic interactions are expressed as rules -not as reactions as in the traditional approach using differential equations or Petri nets, or as agent-centric interactions as in process-algebraic approaches [9].

As we will see below, the presentation of the scheme dynamics as a set of rules not only allows for a crisp description, but also allows one to derive the scheme by a succession of rule refinements, starting from a very basic model. We believe this derivation illuminates the function of our interaction pattern and demonstrates the relevance of refinement as an explanatory force (see Ref. [10] for more examples, and a rigourous account of refinement).

We postpone the discussion of the significance of the notion of refinement, and how one can think of it in relation to the actual evolution of protein networks, to the concluding remarks, and start with an informal presentation of our representational language Kappa (§2), and an outline of the paper (§3).

[1] A model of epigenetic information maintenance, unpublished work in collaboration with Arndt Benecke.

2 Kappa, Briefly

Kappa is a fully implemented stochastic calculus of binding and modification that addresses the modelling of fine-grained regulation in biological systems.[2] The unique construct known to Kappa is called an agent (usually meant to represent a protein), a name (eg identifying the corresponding protein) and a set of sites (eg the protein binding domains and modifiable amino-acids). Each site can carry a value, generally used to represent post-translational modifications. The set of values taken by the sites of an agent is commonly called the agent internal state, and an event that modifies the agent internal state is called a modification. Agents can also use sites to bind other agents -with the key restriction that a site can be used to bind at most one other site at a time. Often the modification of an agent results from another agent binding and modifying it. This is a natural way to represent, for instance, the basic interaction of a substrate with a kinase (as in the example right below).

Events such as binding, unbinding, and modification are described by rules specifying under which condition they may happen. This places Kappa in the family of rule-based modelling languages; another member of this small family is the BNG language. Rules have rates, and following the mass action principle, the likelihood that a given rule applies to a particular system x is proportional to the number of its instances in x multiplied by its rate, aka the rule's activity. The accompanying time advance is on average inverse to the cumulated activity of all rules operating on x. So Kappa is a quantitative framework. The particular case where agents have no sites, and therefore cannot be bound or modified is equivalent to stochastic Petri nets (aka flat chemical reactions, or multiset rewriting). In the presence of sites however, the underlying rewriting theory is richer and is best seen as a kind of stochastic graph rewriting (this point of view is developed in Ref. [10]).

Kappa uses concurrency concepts to dissect the intricate causation mechanisms one finds in protein networks [7]. Tracking down the fate of an agent and accessing the fine causal structure that produces an event of interest is useful as we will see below. In our particular example another interest of using Kappa is that we can derive it by a succession of refinements.

Before we turn to describing our strategy to derive our scheme, let us begin with a simple kinase and substrate example to explain the notation we use and illustrate the notion of refinement.

2.1 An Example

Consider a kinase `K`, and `T` its target, and suppose `T` has two phosphorylatable sites x and y; one can decompose each phosphorylation event in a triplet of elementary ones 1) `K` binds its target `T` at site x or y, 2) `K` may (but need not) phosphorylate the site to which it is bound, 3) `K` dissociates from `T`.

[2] The Kappa implementation includes a graphical interface, is free for academic usage, and can be obtained at `support@plectix.com`.

This translates in the following rules (three per sites in T):

$$\begin{aligned}
&\mathtt{K}(a), \mathtt{T}(x) \rightarrow \mathtt{K}(a^1), \mathtt{T}(x^1)\\
&r_1 := \mathtt{K}(a^1), \mathtt{T}(x^1) \rightarrow \mathtt{K}(a), \mathtt{T}(x)\\
&\mathtt{K}(a^1), \mathtt{T}(x^1_u) \rightarrow \mathtt{K}(a^1), \mathtt{T}(x^1_p)\\
&\mathtt{K}(a), \mathtt{T}(y) \rightarrow \mathtt{K}(a^1), \mathtt{T}(y^1)\\
&\mathtt{K}(a^1), \mathtt{T}(y^1) \rightarrow \mathtt{K}(a), \mathtt{T}(y)\\
&r_2 := \mathtt{K}(a^1), \mathtt{T}(y^1_u) \rightarrow \mathtt{K}(a^1), \mathtt{T}(y^1_p)
\end{aligned}$$

In the textual notation we are using here, internal states are shown as subscripts u (unphosphorylated) and p (phosphorylated) to the sites they are attached to; bonds are represented as shared superscripts across agents to indicate the two endpoints of a link. The number chosen to name a link is irrelevant, as long as it is unique. A double arrow indicates a reversible rule. The left hand side of a rule specifies a condition to trigger the rule, while the right hand side specifies the various changes occurring as a consequence of applying the rule. Thus the third reaction says that when K is bound to T at x, x may be phosphorylated.

Importantly, the sites of an agent need not all be present in a rule, eg in the first rule above, T does not mention y. Likewise, if a site is mentioned at all, its internal state may be left unspecified, eg in the same first rule one does not say whether x in T is phosphorylated or not. This is the trivial but crucial *'don't care, don't write'* convention: only the conditions bearing on the application of a rule need to be represented. Not to put to fine a point on it, the agility of the rule-based approach relies entirely on the ability to trigger events based on partial conditions, an ability that ordinary methods do not have, and process-algebraic methods only partially have (for a more extensive account of the rule-based accrued flexibility see Ref. [11]). Technically partial matches rely on a suitable notion of site graph morphism [10].

2.2 Variants

Another point worth noticing is that Kappa's fine-grained notation forces one to make one's mechanistic choices explicit. Let us consider a few possible variants (among many) of the rules above:

$$\begin{aligned}
&\mathtt{K}(a^1), \mathtt{T}(x^1, y_u) \rightarrow \mathtt{K}(a^1), \mathtt{T}(x^1, y_p)\\
&r'_1 := \mathtt{K}(a^1), \mathtt{T}(x^1_p) \rightarrow \mathtt{K}(a), \mathtt{T}(x_p)\\
&r'_2 := \mathtt{K}(a^1), \mathtt{T}(x^?_p, y^1_u) \rightarrow \mathtt{K}(a^1), \mathtt{T}(x^?_p, y^1_p)
\end{aligned}$$

The first rule extends the modification capacities of K by allowing it to modify T at site y while being bound at x (when y is free); one sometimes say K is processive in this case. The second rule r'_1 allows K to dissociate from T at x only when x is modified. The third rule r'_2 forces the modification order, that is to say y can only be modified after x is; it introduces a new notation, using a ? superscript on x to mean that x can be free or not -all that counts is that x is modified.

One can combine the above variants, and exchange the roles of x, and y to obtain a rather large set of variations on what is presumably the simplest scheme,

ie covalent modification. These various choices are just as key to the dynamics as rule rates are. As said in the introduction access to this fine-grained description is crucial to the definition of a scheme.

The variant rules r'_1, r'_2 have in common that they are more specific than respectively r_1, and r_2. For instance, r'_1 now ensures that K does not let go of its substrate too early, whereas this was possible according to the original dissociation rule r_1. By setting a high rate r'_1 one could also make sure it does let go of the substrate fast when it has been modified. (Presumably one will find cases where selection has led to the evolution of such smart enzymes.)

The substitution of r_1 with the smarter r'_1, or r_2 with r'_2 are simple examples of *refinement*, whereby a rule is replaced with one or more rules that are more specific than the initial rule. While working out in full generality when a refinement is neutral [10], ie when it preserves the underlying dynamics, is not easy, for the simple refinements we need here, an intuitive understanding of the notion will be enough.

3 Outline

Now that we have a better idea of the language we will use, let us explain how we organise the derivation and study of our pattern in the next section (§4).

We first consider a model where:

- a *target* agent $\mathtt{T_1}(k)$ has one site k which can be in one of two internal states, written $\mathtt{T_1}(k_{yes})$, and $\mathtt{T_1}(k_{no})$ (as in §2, internal states are shown as subscripts);
- damage happens to $\mathtt{T_1}$ which means its unique site can spontaneously switch or decay from the 'yes' to the 'no' state: $\mathtt{T_1}(k^?_{yes}) \to \mathtt{T_1}(k^?_{no})$
- damage can be subsequently repaired by an agent K.

(Remind that the superscript ? in the rule above indicates that this event may happen irrespective of whether k is bound or free.)

In order to calibrate the model and measure how well it is doing, we observe how many T_1s are free in the 'yes' state, and how many of these are 'sleeping' -meaning they are in the correct 'yes' state but still bound to K. It is rather easy then to build a basic model that works to our satisfaction in terms of efficiency, ie how fast $\mathtt{T_1}$ is repaired, and this concludes our first step.

The next step consists in perturbing the initial model by introducing a variant $\mathtt{T_2}$ of our initial target agent $\mathtt{T_1}$. This new agent $\mathtt{T_2}$ will have the same rules as $\mathtt{T_1}$ except for repair. Specifically, we will suppose that in order to repair $\mathtt{T_2}$, K needs to be bound to a helper agent H. Intuitively one can think of $\mathtt{T_2}$ as being more fragile or needing more work to be put back in the correct 'yes' state. Obviously, if no rule could sense the $\mathtt{T_1}$, $\mathtt{T_2}$ distinction the behaviour would be unchanged, but precisely the postulated fragility of $\mathtt{T_2}$ introduces such a distinction.

As we will see, this second step can presented by ways of refinement. Under quite general conditions it will break down the repair mechanism, the reason for this being that Hs are distributed uniformly across Ks and will have no reason to

be where they are needed, ie with these particular K$_s$ that happen to bind a T$_2$ and not a T$_1$. In fact, the smaller the fraction of T$_2$ and the more their necessary helpers H will be 'diluted'.

So the next and last step aims at repairing the repair system. One cannot order an H to go where it is needed, however it is entirely possible to trap it there and have it stay longer. To do so, it suffices to refine the rule in charge of the dissociation of the KH bond and modulate its rate depending on whether K is bound to a T$_2$ (low rate) or not (high rate, including the case K is not bound to a target). With this counter-refinement one obtains the pattern we are interested in. No central scheduling mechanism directs H$_s$ to T$_2$s, the refinement just makes the places in need of an H stickier. Yet this purely local trick works just as well as a centralised one would. It is an altogether different (and interesting) question to know whether this construct is also something one can engineer *in vivo*.

4 The Scheme

With our plan in place we can turn to the next section. All the code presented there is written in the syntax of the tool we use to examine the behaviour of our succession of models, and can be used as is. For this to be possible we switch now to a pure ascii notation, whereas so far we had used more mathematical notations with internal states as subscripts and bindings as exponents. Hopefully the slight change of syntax will feel natural.

4.1 A Simple Repair Model

To begin with we have three agents K(t,h), H(k), T$_1(k)$ with the interaction rules shown below. Stochastic rule rates are indicated on the right of each rule.[3]

```
K(t!1),T1(k!1) -> K(t),T1(k) @ 100
K(t),T1(k) -> K(t!1),T1(k!1) @ 1
K(t!1),T1(k~no!1) -> K(t!1),T1(k~yes!1) @ 50

T1(k~yes?) -> T1(k~no?) @ 1

K(h!1),H(k!1) -> K(h),H(k) @ 500
K(h),H(k) -> K(h!1),H(k!1) @ 10
```

One recognises the first three rules as a simple modification triplet, where neither the association, nor the dissociation rule are smart in the sense of the preceding discussion (§2.2). The fourth rule expresses T$_1$'s spontaneous decay. At this stage H plays no role, and is just a passive bystander that may bind K as specified in the last two rules. Not however, while the KH binding has the same equilibrium constant has that of the KT one, we have this binding more labile by using higher on- and off-rates. This will play a role in the implementation of our scheme.

[3] Remind that a stochastic bimolecular rate γ is related to its intensive deterministic analogue as $\gamma AV = k$ where V is a volume and A is Avogadro's number.

As initial conditions we pick:

```
%init: 10 * (K(h,t))
%init: 10 * (H(k))
%init: 100 * (T1(k~yes))
```

As for observables, meaning the objects that we want the simulation to keep track of, as said in §3, we pick the free $\mathtt{T_1}(k_{yes})$, the sleeping ones $\mathtt{T_1}(k^-_{yes})$ where $\mathtt{T_1}$ is bound on k and yet in a proper internal state, and $\mathtt{K}(h\text{-})$ the number of Ks bound to an H -an observable which allows us to monitor the saturation of K on its H side (of which more later):

```
%obs: T1(k~yes)
%obs: T1(k~yes!_)
%obs: K(h!_)
```

This completes the definition of our first model and with this choice of parameters, we find that at steady state about 60% of the $\mathtt{T_1}$s are repaired and free (Fig. 1). The remainder of the $\mathtt{T_1}$s can be considered as being under repair, including those 5% $\mathtt{T_1}$s that are already repaired but not yet released by K.

To ease the reading by reducing variance, copy numbers are rescaled by a factor of 10 in all plots (and bimolecular rates compensated accordingly).

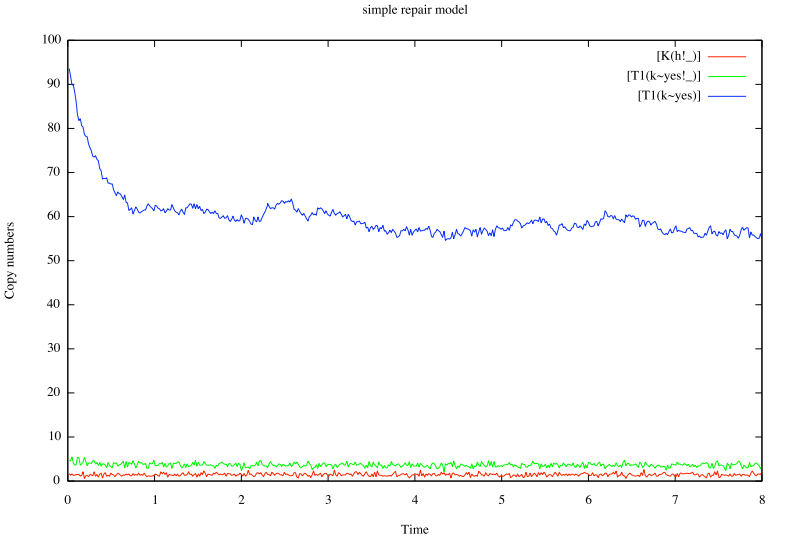

Fig. 1. A run of the simple repair model (rescale 10, 500 data points): about 60% of the $\mathtt{T_1}$s are free and in the 'yes' state; about another 5% are yet to be released; finally about 30% of the Ks are bound to an H

4.2 Introducing $\mathtt{T_2}$

We now take a strict duplicate of $\mathtt{T_1}$ which we call $\mathtt{T_2}$ with identical rules. To do this we could use a refinement introducing a fictitious site of $\mathtt{T_1}$ with an internal

state telling whether the agent is a T_1, or a T_2. (In general, one can encode agent names as internal states that no rule can ever change.)

Equivalently, we just copy the rules:

```
K(t!1),T2(k!1) -> K(t),T2(k) @ 100
K(t),T2(k) -> K(t!1),T2(k!1) @ 1
K(t!1),T2(k~no!1) -> K(t!1),T2(k~yes!1) @ 50

T2(k~yes?) -> T2(k~no?) @ 1
```

We also need to make room for our variant agent in the new initial state (`K`, `H` remain as before at 10 each). We choose here to divide evenly the population into 50% of T_1s and T_2s.

```
%init: 50 * (T1(k~yes))
%init: 50 * (T2(k~yes))
```

We add the following new observable

```
%obs: T2(k~yes)
```

As expected (Fig. 2) nothing changes and the evolution of T_2 is similar to that of T_1 modulo a rescale (inducing somewhat wider fluctuations).

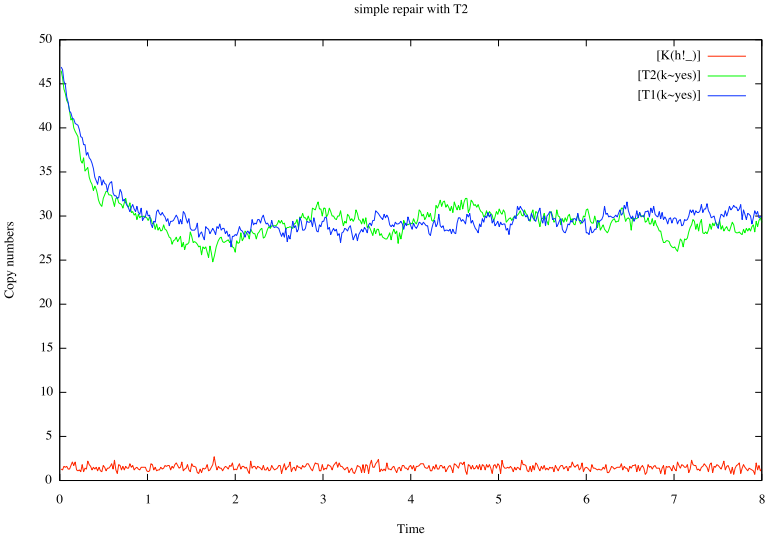

Fig. 2. The repair model with T_2 added (rescale 10, 500 data points): since no rule distinguishes T_1 and T_2: nearly 60% of the T_2s are free and repaired at steady state just as for the T_1s; one sees wider fluctuations because of the population divide

4.3 T_2's Specific H-Mediated Repair

Now is the time where we change our rule set significantly as we replace the above T_2 repair rule with the following:

```
K(t!1,h!_),T2(k~no!1) -> K(t!1,h!_),T2(k~yes!1) @ 50
```

whereby we specify that in order for K to be able to repair T_2, K must be bound to the helper agent H. This constitutes a (non neutral) refinement (the other case where K is free on h is assigned a zero rate). Note that in the actual formulation of the rule, it suffices to state that K is bound on its h site, since only H binds there. This convenient abbreviation would not work if other agents could bind K at h.

As a result we observe a sharp degradation of T_2's repair (Fig. 3).

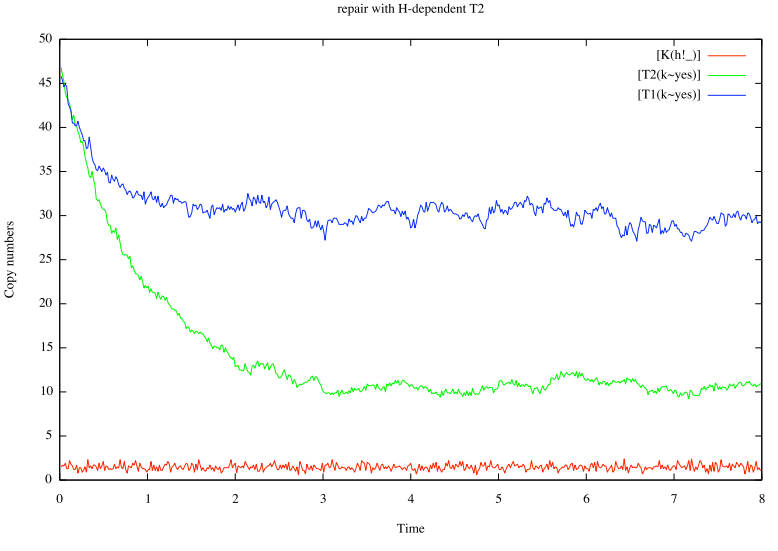

Fig. 3. The third repair model where T_2's repair is H-dependent (rescale 10, 500 data points): less than 20% of the T_2s are in the 'yes' state

Importantly, the behaviour obtained here depends on whether K is *saturated* by H, or in other words on how high the probability that a given K is bound to a T is. To monitor this probability we have added the number of bound Hs in each plot. This probability depends only on the equilibrium dissociation rate of the K and H binding and their initial copy numbers H and K. If it is 1, then T_1 and T_2 are again indistinguishable, since T_2 only behaves differently when H is absent, so T_2 will be repaired just as well as T_1. If it is 0, then T_2 will never be repaired. With our specific choices one has roughly 20% of the K bound to an H, so K is far from being saturated on its H binding site. Even in such intermediate conditions $T_2(k_{yes})$ goes down steeply -which will make the rescue (to come next) more spectacular. One might say that Hs do not find T_2s, and there is a 'stochastic dilution' effect.

4.4 Repairing T_2's Repair

To counter the stochastic dilution of the repair process all one needs is a refinement of the KH dissociation rule. Specifically, we replace that unconditional dissociation rule with the following three rules:

```
K(h!1,t),H(k!1) -> K(h,t),H(k) @ 1000
K(h!1,t!2),H(k!1),T1(k!2) -> K(h,t!2),H(k),T1(k!2) @ 1000
K(h!1,t!2),H(k!1),T2(k!2) -> K(h,t!2),H(k),T2(k!2) @ 0.001
```

We are trying to trap H when a T_2 stands on the other site of K. Thus H will reside longer where it is needed. In the other cases, that is when K is either free on t, or bound to a T_1, we increase the dissociation rate so that T_2s do not linger. The fact that we have chosen early on high on- and off-rates for the KH binding enhances the efficiency of this strategy, since H will explore quickly its potential partners until it finds one that is engaged in the repair of a T_2.

With the above refinement and rate manipulation, one obtains a far more efficient repair rate for T_2, as one can see on Fig. 4. It is important to note that this is not obtained by pushing the KH binding into saturation; indeed, as one can also see, the occupancy rate of Hs has not increased.

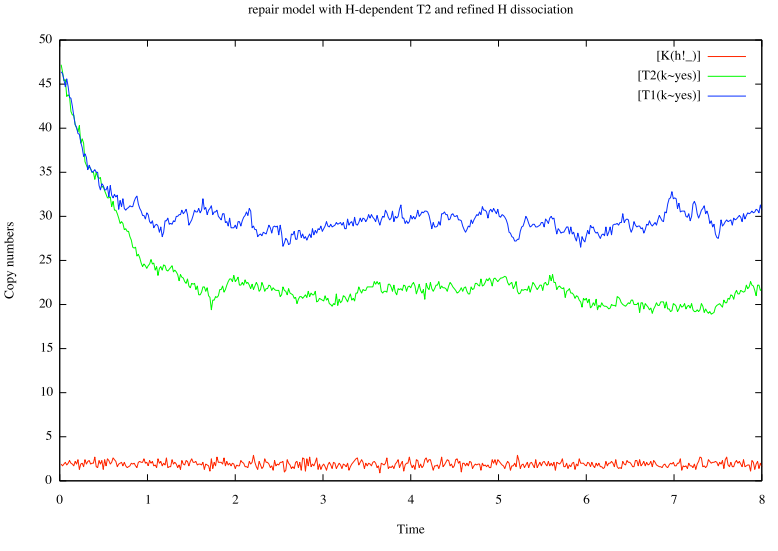

Fig. 4. The last repair model obtained by refining KH dissociation: nearly 50% of the T_2s are free and repaired, which is better than before (Fig. 3), and nearly as good as in the original model (Fig. 2). Take note that there is no significant increase in the H binding to K (lower red curve).

In a biological situation, presumably T_2 will have a function which it can only perform when in its correct 'yes' state, and the additional availability afforded by the refinements above could make all the difference.

5 Conclusion

In the repair scheme we have presented, one has a repair agent K which can recognise different targets, and bring a helper H to to participate in the process for some specific targets. The question is how to get the said helper where it is needed. An easy solution would be to saturate K, in a manner vaguely reminiscent of a Swiss knife, but this begs the question of whether there is perhaps a more elegant and parsimonious solution, where the knife would self-assemble only where and when needed. Indeed, there is one which we have shown in this note, and which one could compare to a stochastic ratchet. This pattern of interaction could extend further than the particular confines of the problem we meant to elucidate in this note -clearly.

With the careful derivation of this scheme, we have made a numerical case for our initially informal intuition, ie we have shown it made numerical sense. This is one of the traditional role of modelling in biology, to strengthen one's assumptions by putting them to numerical test. This we have done purely by simulations, and it would be very interesting to pursue this investigation with a more mathematically grounded approach. Perhaps with a set suitable simplifying assumptions one could have an entirely analytical approach of the problem, which would greatly help to understand how the several parameters at play (rates and copy numbers) impinge on the efficiency of our scheme. In particular, with a purely numerical construction as the one we offered here, it is unclear how the scheme behaves if one changes the fractions of the various repair targets.

As said in the introduction, this scheme was placed in combination with others in a simple model of an aspect of epigenetic repair. In this larger model the helper H stochastic dilution is even more of a problem since non repairing the target on time compromises a temporary memory and leads to permanent mistakes, not just inefficiencies. An interesting consequence of the above investigation is that we have an example where very labile binding (meaning with high off- and on-rates) of the repair agent K to its target is crucial. It has already been observed that key dissociation rates in epigenetic repair are higher than would seem reasonable. It may be that one will observe labile fast diffusing agents as well.

Finally, another point worth noticing is that the scheme was derived by a succession of refinements. In so doing we may also have given some substance to the idea that this scheme can be rather easily 'evolved' by variation and subsequent selection. This is a question that we wish to return to in the future.

References

1. Goldbeter, A., Koshland, D.: An Amplified Sensitivity Arising from Covalent Modification in Biological Systems. Proceedings of the National Academy of Sciences 78(11), 6840–6844 (1981)
2. Lisman, J.E.: A mechanism for memory storage insensitive to molecular turnover: a bistable autophosphorylating kinase. Proc. Natl. Acad. Sci. U S A 82(9), 3055–3057 (1985)

3. Alon, U.: Network motifs: theory and experimental approaches. Nat. Rev. Genet. 8(6), 450–461 (2007)
4. Wall, M.E., Dunlop, M.J., Hlavacek, W.S.: Multiple functions of a feed-forward-loop gene circuit. J. Mol. Biol. 349(3), 501–514 (2005)
5. Mazurie, A., Bottani, S., Vergassola, M.: An evolutionary and functional assessment of regulatory network motifs. Genome Biol. 6(4), R35 (2005)
6. Yeates, T.O., Beeby, M.: Proteins in a small world. Science 314(5807), 1882–1883 (2006)
7. Danos, V., Feret, J., Fontana, W., Harmer, R., Krivine, J.: Rule-based modelling of cellular signalling. In: Caires, L., Vasconcelos, V.T. (eds.) CONCUR 2007. LNCS, vol. 4703, pp. 17–41. Springer, Heidelberg (2007)
8. Blinov, M.L., Faeder, J.R., Goldstein, B., Hlavacek, W.S.: A network model of early events in epidermal growth factor receptor signaling that accounts for combinatorial complexity. BioSystems 83, 136–151 (2006)
9. Priami, C., Regev, A., Shapiro, E., Silverman, W.: Application of a stochastic name-passing calculus to representation and simulation of molecular processes. Information Processing Letters (2001)
10. Danos, V., Feret, J., Fontana, W., Harmer, R., Krivine, J.: Rule-based modelling, symmetries, refinements. In: Fisher, J. (ed.) FMSB 2008. LNCS (LNBI), vol. 5054, pp. 103–122. Springer, Heidelberg (2008)
11. Danos, V.: Agile Modelling of Cellular Signalling. Computation in Modern Science and Engineering 2, Part A 963, 611–614 (2007)