A Logical Framework for Systems Biology

Elisabetta De Maria, Joëlle Despeyroux, Amy Felty

A Logical Framework for Systems Biology

Elisabetta de Maria[1], Joëlle Despeyroux[2], and Amy P. Felty[3]

[1] Laboratoire I3S, University of Nice - Sophia-Antipolis, Sophia-Antipolis, France
edemaria@i3s.unice.fr
[2] INRIA and CNRS, Laboratoire I3S, UNS, Sophia-Antipolis, France
joelle.despeyroux@inria.fr
[3] School of Electrical Engineering and Computer Science, University of Ottawa
Ottawa, Canada, afelty@eecs.uottawa.ca

Abstract. We propose a novel approach for the formal verification of biological systems based on the use of a modal linear logic. We show how such a logic can be used, with worlds as instants of time, as an unified framework to encode both biological systems and temporal properties of their dynamic behaviour. To illustrate our methodology, we consider a model of the P53/Mdm2 DNA-damage repair mechanism. We prove several properties that are important for such a model to satisfy and serve to illustrate the promise of our approach. We formalize the proofs of these properties in the Coq Proof Assistant, with the help of a Lambda Prolog prover for partial automation of the proofs.

1 Introduction

In this paper, we consider the question of reasoning about biological systems in a modal linear logic. We show that a new logic, called Hybrid Linear Logic (HyLL) developed by the second author in joint work with K. Chaudhuri [8, 14], is particularly well-suited to this purpose. HyLL provides a unified framework to encode biological systems, to express temporal properties of their dynamic behaviour, and to prove these properties. By constructing proofs in the HyLL logic, we directly witness reachability as logical entailment. This approach is in contrast to most current approaches to applying formal methods to systems biology, which generally encode biological systems either in a dedicated programming language or in differential equations, express properties in a temporal logic, and then verify these properties against some form of traces built using an external simulator. In the next subsection, we review in some detail the state of the art of such approaches, in order to further situate and motivate our new approach. In Sect. 1.2, we motivate our choice of linear logic in general and HyLL in particular. Then in Sect. 1.3, we further outline our contributions as well as the overall organization of the rest of the paper.

1.1 Formal Methods for Systems Biology

Computational systems biology provides a variety of methods for understanding the structure of biological systems and for studying their dynamics, that is, the temporal evolution of the involved entities.

To capture the qualitative nature of dynamics, Thomas introduced a Boolean approach for regulatory networks (an entity is present or absent) [32] and subsequently generalized it to multivalued levels of concentration [33]. Contrary to Petri nets, which are based on synchronous updating techniques [29], Thomas' discrete models are asynchronous. Other purely qualitative approaches are π-calculus [31], bio-ambients [30], and reaction rules [13, 7].

To describe the dynamics from a quantitative point of view, ordinary or stochastic differential equations are heavily used. More recent approaches include hybrid Petri nets [22] and hybrid automata [1, 5], piecewise linear equations [24], stochastic π-calculus [28], and rule-based languages with continuous/stochastic dynamics such as Kappa [13].

The Biochemical Abstract Machine Biocham [15] is a framework that allows the description of a biochemical system in terms of reaction rules and the interpretation of it at different levels of abstraction, by either an asynchronous Boolean transition system (Boolean semantics), a continuous time Markov chain (stochastic semantics), or a system of ordinary differential equations over molecular concentrations (differential semantics).

One of the most common approaches to the formal verification of biological systems is model checking [11]. Model checking allows one to verify desirable properties of a system by an exhaustive enumeration of all the states reachable by the system. In order to apply such a technique, the biological system should be encoded as a finite transition system and relevant system properties should be specified using propositional temporal logic. Formally, a transition system over a set AP of atomic propositions is a tuple $M = (Q, T, L)$, where Q is a finite set of states, $T \subseteq Q \times Q$ is a total transition relation (that is, for every state $q \in Q$ there is a state $q' \in Q$ such that $T(q, q')$), and $L : Q \to 2^{AP}$ is a labeling function that maps every state into the set of atomic propositions that hold at that state.

Temporal logics are formalisms for describing sequences of transitions between states. The computation tree logic CTL* allows one to describe properties of computation trees. Its formulas are obtained by (repeatedly) applying Boolean connectives, *path quantifiers*, and *state quantifiers* to atomic formulas. The path quantifier A (resp., E) can be used to state that all paths (resp., some path) starting from a given state have some property. The state quantifiers are X (next time), F (sometimes in the future), G (always in the future), and U (until). The branching time logic CTL is a fragment of CTL* that allows quantification over the paths starting from a given state. Unlike CTL*, it constrains every state quantifier to be immediately preceded by a path quantifier. The linear time logic LTL is another known fragment of CTL* where one may only describe events along a single computation path. The Probabilistic Computation Tree Logic PCTL quantifies the different paths by replacing the E and A modalities of CTL by probabilities.

In Biocham, CTL, LTL, and a fragment of PCTL with numerical constraints are used in the three semantics of reaction models, respectively, in the boolean semantics, in the differential semantics and in the stochastic semantics.

Given a transition system $M = (Q, T, L)$, a state $q \in Q$, and a temporal logic formula φ expressing some desirable property of the system, the *model checking problem* consists of establishing whether φ holds at q or not, namely, whether $M, q \models \varphi$. Another formulation of the model checking problem consists of finding all the states $q \in Q$ such that $M, q \models \varphi$. Observe that the second formulation is more general than the first one.

There exist several tools for checking if a finite state system verifies a given CTL, LTL, or PCTL formula, e.g., NuSMV [10], SPIN [23], and PRISM [20].

In contrast to the above approaches, in our new technique we encode both biological systems and temporal properties in HyLL, and prove that the properties can be derived from the system. We focus on Boolean systems and in this case a time unit corresponds to a transition in the system. We believe that discrete modeling is crucial in systems biology because it allows taking into account some phenomena that have a very low chance of happening (and could thus be neglected by differential approaches), but which may have a strong impact on system behaviour.

1.2 Linear Logic

Linear Logic (LL) [19] is particularly well suited for describing state transition systems. LL has been successfully used to model such diverse systems as: Petri nets, CCS, the π-calculus [6, 26], concurrent ML [6], security protocols [4], multiset rewriting, and games.

In the area of biology, for example, a rule of activation (e.g., a protein activates a gene or the transcription of another protein) can be modeled by the following LL axiom:

$$\texttt{active}(a, b) \stackrel{\text{def}}{=} \texttt{pres}(a) \rightarrow (\texttt{pres}(a) \otimes \texttt{pres}(b)).$$

The formula $\texttt{active}(a, b)$ describes the fact that a state where a is present ($\texttt{pres}(a)$ is true) can evolve into a state where both $\texttt{pres}(a)$ and $\texttt{pres}(b)$ are true.

Propositions such as $\texttt{pres}(a)$ are called *resources*, and a rule in the logic can be viewed as a rewrite rule from a set of resources into another set of resources, where a set of resources describes a state of the system. Thus, a particular state transition system can be modeled by a set of rules of the above shape. The rules of the logic then allow us to prove some desired properties of the system, such as, for example, the existence of a stable state.

However, linear implication is timeless: there is no way to correlate two concurrent transitions. If resources have lifetimes and state changes have temporal, probabilistic or stochastic *constraints*, then the logic will allow inferences that may not be realizable in the system being modeled. This was the motivation of the development of HyLL, which was designed to represent constrained transition systems.

1.3 Contributions and Organization

In this work, we present some first applications of HyLL to systems biology. We present HyLL in Sect. 2 and the overall approach to the application domain in Sect. 3. We choose a simple yet representative biological example concerning the DNA-damage repair mechanism based on proteins p53 and Mdm2, and present and prove several properties of this system (Sect. 4). We fully formalize these proofs in a theorem prover we have implemented in the Coq Proof Assistant [2] and λProlog [27] (Sect. 5). This prover is designed to both reason in HyLL and to formalize meta-theoretic properties about it.

We discuss the merits and eventual drawbacks of this new approach compared to approaches using temporal logic and model checking. To better illustrate the correspondence with such approaches, we also present in some detail the encoding of temporal logic operators in HyLL (Sect. 6).

We conclude and discuss future work in Sect. 7.

An electronic appendix (`fmmb-eappendix.pdf`) as well as the formal proofs and a report describing our informal proofs in more detail can be found at `www.eecs.uottawa.ca/~afelty/fmmb14/`. All references to the appendix in this paper point to this web site.

2 A Hybrid Linear Logic

HyLL is a conservative extension of intuitionistic first-order linear logic (LL) [19] where the truth judgements are parameterized on a *constraint domain*: $A \mathbin{@} w$ stands for the truth of A under constraint w. A typical example of such a judgement is "A is true at time t", or "with probability p."

2.1 HyLL Syntax

Like in the linear logic LL, propositions are interpreted as *resources* which may be composed into a *state* using the usual linear connectives, and the linear implication ($\multimap$) denotes a transition between states. The world label w of a judgement $A \mathbin{@} w$ represents a constraint on states and state transitions; particular choices for the worlds produce particular instances of HyLL. The common component in all the instances of HyLL is the proof theory, which is fixed once and for all. The minimal requirement on the kinds of constraints that HyLL can deal with is defined as follows:

Definition 1. *A* constraint domain $\mathcal{W}$ *is a monoid structure* $\langle W, ., \iota \rangle$*. The elements of* W *are called* worlds*, and the partial order* $\preceq : W \times W$ *—defined as* $u \preceq w$ *if there exists* $v \in W$ *such that* $u.v = w$ *—is the* reachability relation *in* $\mathcal{W}$*.*

The identity world ι is $\preceq$-initial and is intended to represent the lack of any constraints. Thus, the ordinary first-order linear logic is embeddable into any

instance of HyLL by setting all world labels to the identity. A typical and simple example of constraint domain is $\mathcal{T} = \langle \Re^+, +, 0 \rangle$, or $\langle I\!N, +, 0 \rangle$, representing instants of time.

Atomic propositions are written using lowercase letters $(p, q, ...)$ applied to a sequence of *terms* $(s, t, \ldots)$, which are drawn from an untyped term language containing constants $(c, \ldots)$, term variables $(x, y, \ldots)$ and function symbols $(f, g, ...)$ applied to a list of terms $(\vec{t})$. Non-atomic propositions are constructed from the connectives of first-order intuitionistic linear logic and the two hybrid connectives *satisfaction* (`at`), which states that a proposition is true at a given world $(w, \iota, u.v, \ldots)$, and *localization* $(\downarrow)$, which binds a name for the (current) world the proposition is true at. The following grammar summarizes the syntax of HyLL terms (t), and propositions (A, B).

$$
\begin{array}{lll}
t & ::= & c \mid x \mid f(\vec{t}) \\
A, B & ::= & p(\vec{t}) \mid A \otimes B \mid \mathbf{1} \mid A \multimap B \mid A \,\&\, B \mid \top \mid A \oplus B \mid \mathbf{0} \mid !A \mid \\
 & & \forall x.\ A \mid \exists x.\ A \mid (A \ \texttt{at}\ w) \mid \downarrow u.\ A \mid \forall u.\ A \mid \exists u.\ A
\end{array}
$$

Note that world u is bound in the propositions $\downarrow u.A$, $\forall u.\ A$ and $\exists u.\ A$.

World variables cannot be used in terms, and neither can term variables occur in worlds; this restriction is important for the modular design of HyLL because it keeps purely logical truth separate from constraint truth. We let α range over variables of either kind. Note that $\downarrow$ and `at` commute freely with all non-hybrid connectives [8].

2.2 Sequent Calculus for HyLL

We present the syntax of hybrid logic in a sequent calculus style [18], using sequents of the form $\Gamma; \Delta \vdash C \ @\ w$ where Γ and Δ are sets of judgements of the form $A \ @\ w$, with Δ being moreover a *multiset*. Γ is called the *unrestricted context*: its hypotheses can be consumed any number of times. Δ is a *linear context*: every hypothesis in it must be consumed singly in the proof. Note that in a judgement $A \ @\ w$ (as in a proposition $A \ \texttt{at}\ w$), w can be any expression in $\mathcal{W}$, not only a variable.

The main set of inference rules is in Fig. 1 (the complete set is in Appendix A). The rules for the first-order quantifiers (omitted) and the exponential ! are completely standard. A brief discussion of the hybrid rules follows. To introduce the *satisfaction* proposition $(A \ \texttt{at}\ u)$ (at any world w) on the right, the proposition A must be true in the world u. The proposition $(A \ \texttt{at}\ u)$ itself is then true at any world, not just in the world u. In other words, $(A \ \texttt{at}\ u)$ carries with it the world at which it is true. Therefore, suppose we know that $(A \ \texttt{at}\ u)$ is true (at any world v); then, we also know that $A \ @\ u$, and we can use this hypothesis (rule "at L"). The other hybrid connective of *localisation*, $\downarrow$, is intended to be able to name the current world. That is, if $\downarrow u.\ A$ is true at world w, then the variable u stands for w in the body A. This interpretation is reflected in its right introduction rule $\downarrow R$. For left introduction, suppose we have a proof of $\downarrow u.\ A \ @\ v$ for some world v. Then, we also know, and thus can use $A[v/u] \ @\ v$.

Note that there are only two structural rules in HyLL. Weakening and contraction are admissible rules. For example:

Judgemental rules

$$\Gamma; p(\vec{t}) \mathbin{@} w \vdash p(\vec{t}) \mathbin{@} w \ [init] \qquad \frac{\Gamma, A \mathbin{@} u; \Delta, A \mathbin{@} u \vdash C \mathbin{@} w}{\Gamma, A \mathbin{@} u; \Delta \vdash C \mathbin{@} w} \ copy$$

Multiplicative

$$\frac{\Gamma; \Delta \vdash A \mathbin{@} w \quad \Gamma; \Delta' \vdash B \mathbin{@} w}{\Gamma; \Delta, \Delta' \vdash A \otimes B \mathbin{@} w} \otimes R \qquad \frac{\Gamma; \Delta, A \mathbin{@} u, B \mathbin{@} u \vdash C \mathbin{@} w}{\Gamma; \Delta, A \otimes B \mathbin{@} u \vdash C \mathbin{@} w} \otimes L$$

$$\Gamma; . \vdash \mathbf{1} \mathbin{@} w \ [\mathbf{1}R] \qquad \frac{\Gamma; \Delta \vdash C \mathbin{@} w}{\Gamma; \Delta, 1 \mathbin{@} u \vdash C \mathbin{@} w} \, 1\, L$$

$$\frac{\Gamma; \Delta, \ A \mathbin{@} w \vdash B \mathbin{@} w}{\Gamma; \Delta \vdash A \to B \mathbin{@} w} \to R \qquad \frac{\Gamma; \Delta \vdash A \mathbin{@} u \quad \Gamma; \Delta', B \mathbin{@} u \vdash C \mathbin{@} w}{\Gamma; \Delta, \Delta', A \to B \mathbin{@} u \vdash C \mathbin{@} w} \to L$$

Additive

$$\Gamma; \Delta \vdash T \mathbin{@} w \ [T\ R] \qquad \Gamma; \Delta, \mathbf{0} \mathbin{@} u \vdash C \mathbin{@} w \ [\mathbf{0}L]$$

$$\frac{\Gamma; \Delta \vdash A \mathbin{@} w \qquad \Gamma; \Delta \vdash B \mathbin{@} w}{\Gamma; \Delta \vdash A \,\&\, B \mathbin{@} w} \ \& R \qquad \frac{\Gamma; \Delta, A_i \mathbin{@} u \vdash C \mathbin{@} w}{\Gamma; \Delta, A_1 \,\&\, A_2 \mathbin{@} u \vdash C \mathbin{@} w} \ \& L_i$$

$$\frac{\Gamma; \Delta \vdash A_i \mathbin{@} w}{\Gamma; \Delta \vdash A_1 \oplus A_2 \mathbin{@} w} \oplus R_i \qquad \frac{\Gamma; \Delta, A \mathbin{@} u \vdash C \mathbin{@} w \quad \Gamma; \Delta, B \mathbin{@} u \vdash C \mathbin{@} w}{\Gamma; \Delta, A \oplus B \mathbin{@} u \vdash C \mathbin{@} w} \ \oplus L$$

Exponentials rules

$$\frac{\Gamma; . \vdash A \mathbin{@} w}{\Gamma; . \vdash !A \mathbin{@} w} \ !R \qquad \frac{\Gamma, A \mathbin{@} u; \Delta \vdash C \mathbin{@} w}{\Gamma; \Delta, !A \mathbin{@} u \vdash C \mathbin{@} w} \ !L$$

Hybrid connectives

$$\frac{\Gamma; \Delta \vdash A \mathbin{@} u}{\Gamma; \Delta \vdash (A \ \mathtt{at}\ u) \mathbin{@} w} \ [\mathtt{at}\ R] \qquad \frac{\Gamma; \Delta, A \mathbin{@} u \vdash C \mathbin{@} w}{\Gamma; \Delta, (A \ \mathtt{at}\ u) \mathbin{@} v \vdash C \mathbin{@} w} \ [\mathtt{at}\ L]$$

$$\frac{\Gamma; \Delta \vdash A[w/u] \mathbin{@} w}{\Gamma; \Delta \vdash \downarrow u.A \mathbin{@} w} \ [\downarrow R] \qquad \frac{\Gamma; \Delta, A[v/u] \mathbin{@} v \vdash C \mathbin{@} w}{\Gamma; \Delta, \downarrow u.A \mathbin{@} v \vdash C \mathbin{@} w} \ [\downarrow L]$$

Fig. 1. (Part of) the sequent calculus for HyLL

Theorem 2 (weakening). *If* $\Gamma; \Delta \vdash C \mathbin{@} w$*, then* $\Gamma, \Gamma'; \Delta \vdash C \mathbin{@} w$*.*

The most important structural properties are the admissibility of the identity and cut theorem; the latter guarantees consistency.

Theorem 3 (identity). $\Gamma, A \mathbin{@} w \vdash A \mathbin{@} w$*.*

Theorem 4 (cut).
1. *If* $\Gamma; \Delta \vdash A \mathbin{@} u$ *and* $\Gamma; \Delta', A \mathbin{@} u \vdash C \mathbin{@} w$*, then* $\Gamma; \Delta, \Delta' \vdash C \mathbin{@} w$
2. *If* $\Gamma; . \vdash A \mathbin{@} u$ *and* $\Gamma, A \mathbin{@} u; \Delta \vdash C \mathbin{@} w$*, then* $\Gamma; \Delta \vdash C \mathbin{@} w$*.*

An example of derived statements, true in every semantics for worlds, is the following:

Proposition 5 (relocalisation). *Any true judgement can be relocated at any time in the future:*

$$\frac{\Gamma; A_1 \ @ \ w_1 \cdots A_k \ @ \ w_k \vdash B \ @ \ v}{\Gamma; A_1 \ @ \ u.w_1 \cdots A_k \ @ \ u.w_k \vdash B \ @ \ u.v}$$

This property is particularly well suited to applications in biology. The interested reader can find proofs and further meta-theoretical theorems about HyLL in [8].

2.3 Some Definitions for Biology

We can define modal connectives in HyLL as follows:

Definition 6 (modal connectives).

$$\Box A \stackrel{\text{def}}{=} \downarrow u.\ \forall w.\ (A \ \mathtt{at} \ u.w) \qquad \Diamond A \stackrel{\text{def}}{=} \downarrow u.\ \exists w.\ (A \ \mathtt{at} \ u.w)$$
$$\delta_v \, A \stackrel{\text{def}}{=} \downarrow u.\ (A \ \mathtt{at} \ u.v) \qquad \dagger \, A \stackrel{\text{def}}{=} \forall u.\ (A \ \mathtt{at} \ u)$$

The connective δ represents a form of delay. Note its derived right rule:

$$\frac{\Gamma \vdash A \ @ \ w.v}{\Gamma \vdash \delta_v \, A \ @ \ w} \ [\delta R]$$

The proposition $\delta_v \, A$ thus stands for an *intermediate state* in a transition to A. Informally it can be thought to be "v before A". The modally unrestricted proposition $\dagger \, A$ represents a resource that is consumable in any world; it is mainly used to make transition rules applicable at all worlds.

Oscillation is one of the typical properties of interest in biological systems (illustrated here by Property 1 in Sect. 4.3). In our logic, we can define one oscillation between A and B, with respective delays u and v, as follows:

Definition 7 (one oscillation).

$$\mathit{oscillate}_1 \, (A, B, u, v) \stackrel{\text{def}}{=} A \ \ \& \ \ \delta_u (B \ \ \& \ \ \delta_v \, A) \ \ \& \ \ (A \,\&\, B \to 0)$$

Note that the above HyLL proposition closely corresponds to the temporal formula $A \wedge \mathsf{EF}(B \wedge \mathsf{EF} A)$.

Oscillation can be more generally defined by the following proposition in HyLL:

Definition 8 (oscillation).

$$\mathit{oscillate}_h \, (A, B, u, v) \stackrel{\text{def}}{=} \dagger[(A \to \delta_u \, B) \ \ \& \ \ (B \to \delta_v \, A)] \ \ \& \ \ (A \,\&\, B \to 0).$$

However, since oscillation can be considered a meta-level property of the biological systems modeled in HyLL, this property is perhaps more naturally defined as follows:

Definition 9 (oscillation). $\mathit{oscillate} \, (A, B, u, v) \stackrel{\text{def}}{=}$ *for any* w, $(A \ @ \ w \vdash B \ @ \ w.u)$, $(B \ @ \ w.u \vdash A \ @ \ w.u.v)$, *and* $(\vdash A \,\&\, B \to 0 \ @ \ w)$.

2.4 Temporal Constraints

In this paper, we only consider the constraint domain $\mathcal{T} = \langle I\!N, +, 0\rangle$ representing (discrete) instants of time, and we write HyLL[$\mathcal{T}$] for this instantiation of HyLL. Delay (Definition 6) in HyLL[$\mathcal{T}$] represents intervals of time; $\delta_d\, A$ means "A will become available after delay d". For our temporal specifications the notion of addition on times is fundamental.

3 Approach

In this work we take into consideration Boolean models consisting of (i) a set of Boolean variables, (ii) a (partially defined) initial state denoting the presence/absence of (some) variables, and (iii) a set of rules of the form $L_i \Rightarrow R_i$, where the left (resp. right) hand side of the rule L_i (resp. R_i) is the conjunction of a set of predicates concerning the presence/absence of variables. For example, $x_1 \wedge x_2 \Rightarrow \neg x_3$ is a valid rule. This kind of rule can be used to describe state transitions involving control variables or abstract processes. In Biocham [15], they are mostly used to represent biochemical reactions (observe that Biocham rules are more restrictive, they do not express the absence of a variable). In this paper, we take advantage of these rules to express influence rules (e.g. activations and inhibitions) in a biological system.

Observe that, given a Boolean model of this kind, although we do not do it, it is always possible to build a transition system where the set of states is the set of all tuples of Boolean values denoting the presence/absence of the different variables, and a pair of states (s,s') belongs to the relation if and only if there exists a rule i such that s satisfies L_i, s' satisfies R_i, and all the variables not involved in rule i have the same value in s and s'.

If L_i speaks about the presence of a variable x, nothing can be said about the presence/absence of x in s'. In other words, nothing can be inferred about the consumption of variables appearing in the left hand side of rules. If we want to force consumption/not consumption, we have to specify it explicitly. Furthermore, our rules are asynchronous: one rule can be fired at a time. If several rules can be taken from a given state, one of them is non-deterministically chosen. As in Biocham, we choose an asynchronous semantics in order to eliminate the risk of affecting fundamental biological phenomena such as the masking of a relation by another one and the consequent inhibition/activation of biological processes.

To verify whether a Boolean model satisfies a given temporal property, our approach consists of encoding both the model and the property in the HyLL logic and producing a proof. Observe that we do not explicitly build the transition system, we just give a set of variables and rules. Proving temporal properties can result in building a sub-part of the system. There is an analogy with on-the-fly model checking [12], a technique that in many cases avoids the construction of the entire state space of the system (because the property to test guides the construction of the system).

4 Example

In this section we focus on the P53/Mdm2 DNA-damage repair mechanism. P53 is a tumor suppressor protein that is activated in reply to DNA damage. In normal conditions, the concentration of p53 in the nucleus of a cell is weak: its level is controlled by another protein, Mdm2. These two proteins present a loop of negative regulation. In fact, P53 activates the transcription of Mdm2 while the latter accelerates the degradation of the former.

DNA damage increases the degradation rate of Mdm2 so that the control of this protein on P53 becomes weaker and the concentration of p53 can increase. P53 can thus exercise its functions, either stopping the cell cycle to allow DNA repair, or provoking apoptosis, if damage is too heavy.

When Mdm2 loosens its influence on P53, it is possible to observe some oscillations of P53 and Mdm2 concentrations. The answer to a stronger damage is a bigger number of oscillations. In the literature, several models have been proposed to model the oscillatory behaviour of proteins P53 and Mdm2 (see e.g. Ciliberto et al. [9]). In [25] it is shown how a model of the P53/Mdm2 DNA-damage repair mechanism can be exploited in the study of cancer therapies.

4.1 Definition

In the following we propose a simple Boolean model considering the presence/absence of the three variables DNAdam, P53, and MdM2. Initial states are the ones where P53 is absent and Mdm2 is present.

The behaviour of the biological system is specified by the six following rules:

1) Dnadam $\Rightarrow \neg$Mdm2	4) Mdm2 $\Rightarrow \neg$P53
2) $\neg$Mdm2 $\Rightarrow$ P53	5) P53 $\Rightarrow_C \neg$Dnadam
3) P53 $\Rightarrow$ Mdm2	6) $\neg$Dnadam $\Rightarrow$ Mdm2

In rule 5, we use $\Rightarrow_C$ to force consumption, i.e., if P53 is present, after firing the rule both P53 and Dnadam are absent. Note that, if Dnadam is present in the initial state, this rule (that refers to damage repair) can be non-deterministically fired after one, a few, or several P53/MdM2 oscillations. This is consistent with experiments, where the number of oscillations preceding damage repair depends on the damage entity. In the other rules we assume there is no consumption.

4.2 Specification in HyLL

The biological system is modeled in HyLL by a set of axioms of two kinds. First, each rule of the biological system is modeled by a formula in HyLL, as it would be in ordinary linear logic, with the additional use of the delay operator δ_v, making precise the delay taken by the corresponding transition. The domain of world is $\mathcal{T} = \langle I\!N, +, 0\rangle$ and we fix all time delays to 1. Then we add a set of axioms stating some well-definedness conditions and the initial state. To encode the basic Boolean model, we use two predicates: $\texttt{pres}(a)$ (seen earlier) and $\texttt{abs}(a)$ to indicate the presence or absence of variable a. The full model is given in Fig. 2.

Activation/Inhibition Rules For the sake of clarity, we first define generic activation/inhibition actions. These actions can be defined in various ways. A first attempt at describing an activation rule without consumption, for example, might be the following:

$$w\_\texttt{active}(a,b) \stackrel{\text{def}}{=} \texttt{pres}(a) \to \delta_1(\texttt{pres}(a) \otimes \texttt{pres}(b)).$$

Note that in the case where b is present, the above rule does not modify the state. In order to avoid uninteresting proofs, we might alternatively define our activation rule in a more precise way, as follows:

$$s\_\texttt{active}(a,b) \stackrel{\text{def}}{=} \texttt{pres}(a) \otimes \texttt{abs}(b) \to \delta_1(\texttt{pres}(a) \otimes \texttt{pres}(b)).$$

We call the first kind of rules *weak* rules, and the second one *strong* rules. For the sake of completeness, let us mention a third kind of rules, that we might call *useless*:

$$u\_\texttt{active}(a,b) \stackrel{\text{def}}{=} \texttt{pres}(a) \otimes \texttt{pres}(b) \to \delta_1(\texttt{pres}(a) \otimes \texttt{pres}(b)).$$

The *general* form of an activation rule, taking in account the various possible values of b, and our eventual missing knowledge of this, is then the following:

$$\begin{aligned}\texttt{active}(a,b) \stackrel{\text{def}}{=} &(\texttt{pres}(a) \oplus (\texttt{pres}(a) \otimes \texttt{pres}(b)) \oplus (\texttt{pres}(a) \otimes \texttt{abs}(b))) \\ &\to \delta_1(\texttt{pres}(a) \otimes \texttt{pres}(b)).\end{aligned}$$

The properties stated in the present paper will all use the general form of the biological rules, except Property 4, which is only valid for strong rules.

There are further alternatives for the activation/inhibition rules. Let us consider the above strong variant $s\_\texttt{active}(a,b)$. We can define an *activation with consumption* (of the product a) as follows:

$$s\_\texttt{active}_c(a,b) \stackrel{\text{def}}{=} \texttt{pres}(a) \otimes \texttt{abs}(b) \to \delta_1(\texttt{abs}(a) \otimes \texttt{pres}(b)),$$

while a *strong activation* will have an inhibitor effect, in case of absence of a:

$$s\_\texttt{active}_s(a,b) \stackrel{\text{def}}{=} \texttt{abs}(a) \otimes \texttt{pres}(b) \to \delta_1(\texttt{abs}(a) \otimes \texttt{abs}(b)).$$

Our example also uses the corresponding three kinds of rules for the inhibition actions (see Fig. 2). Of course, we could also define activation/inhibition rules accounting for a lack of information concerning consumption.

The System Before giving the complete definition of our system, we need to additionally specify, in each rule, that if a variable is not touched, then its value remains the same in the next state. This is the purpose of the `unchanged` predicate, which in turn leads us to introduce a parameter `vars`, specifying the set of variables of the biological system.

Note that the definition of the `unchanged` predicate relies on the hypothesis (discussed earlier, in Sect. 3) that only one rule of the biological system can fire at a time. Note also the use of the ! and & operators in its definition: this is an intuitionistic predicate.

– *Variables*:

$\texttt{unchanged}(x, w) \stackrel{\text{def}}{=} \; !\,[(\texttt{pres}(x)\ \texttt{at}\ w \to \texttt{pres}(x)\ \texttt{at}\ w.1)\ \&$
$(\texttt{abs}(x)\ \texttt{at}\ w \to \texttt{abs}(x)\ \texttt{at}\ w.1)].$

$\texttt{unchanged}(V, w) \stackrel{\text{def}}{=} \otimes_{x \in V} \texttt{unchanged}(x, w).$

– *Activation*:

$\texttt{active}(V, a, b) \stackrel{\text{def}}{=} (\texttt{pres}(a) \oplus (\texttt{pres}(a) \otimes \texttt{pres}(b)) \oplus (\texttt{pres}(a) \otimes \texttt{abs}(b)))$
$\to \delta_1(\texttt{pres}(a) \otimes \texttt{pres}(b)) \otimes \downarrow u.\ \texttt{unchanged}(V \setminus \{a, b\}, u)).$

– *Activation with consumption*:

$\texttt{active}_c(V, a, b) \stackrel{\text{def}}{=} (\texttt{pres}(a) \oplus (\texttt{pres}(a) \otimes \texttt{pres}(b)) \oplus (\texttt{pres}(a) \otimes \texttt{abs}(b)))$
$\to \delta_1(\texttt{abs}(a) \otimes \texttt{pres}(b)) \otimes \downarrow u.\ \texttt{unchanged}(V \setminus \{a, b\}, u)).$

– *Strong activation*:

$\texttt{active}_s(V, a, b) \stackrel{\text{def}}{=} (\texttt{abs}(a) \oplus (\texttt{abs}(a) \otimes \texttt{pres}(b)) \oplus (\texttt{abs}(a) \otimes \texttt{abs}(b)))$
$\to \delta_1(\texttt{abs}(a) \otimes \texttt{abs}(b)) \otimes \downarrow u.\ \texttt{unchanged}(V \setminus \{a, b\}, u)).$

– *Inhibition*:

$\texttt{inhib}(V, a, b) \stackrel{\text{def}}{=} (\texttt{pres}(a) \oplus (\texttt{pres}(a) \otimes \texttt{pres}(b)) \oplus (\texttt{pres}(a) \otimes \texttt{abs}(b)))$
$\to \delta_1(\texttt{pres}(a) \otimes \texttt{abs}(b)) \otimes \downarrow u.\ \texttt{unchanged}(V \setminus \{a, b\}, u)).$

– *Inhibition with consumption*:

$\texttt{inhib}_c(V, a, b) \stackrel{\text{def}}{=} (\texttt{pres}(a) \oplus (\texttt{pres}(a) \otimes \texttt{pres}(b)) \oplus (\texttt{pres}(a) \otimes \texttt{abs}(b)))$
$\to \delta_1(\texttt{abs}(a) \otimes \texttt{abs}(b)) \otimes \downarrow u.\ \texttt{unchanged}(V \setminus \{a, b\}, u)).$

– *Strong inhibition*:

$\texttt{inhib}_s(V, a, b) \stackrel{\text{def}}{=} (\texttt{abs}(a) \oplus (\texttt{abs}(a) \otimes \texttt{pres}(b)) \oplus (\texttt{abs}(a) \otimes \texttt{abs}(b)))$
$\to \delta_1(\texttt{abs}(a) \otimes \texttt{pres}(b)) \otimes \downarrow u.\ \texttt{unchanged}(V \setminus \{a, b\}, u)).$

– *Well definedness*:

$\texttt{well\_defined}_0(V) \stackrel{\text{def}}{=} \forall a \in V.\ [\texttt{pres}(a) \otimes \texttt{abs}(a) \to 0].$

$\texttt{well\_defined}_1(V) \stackrel{\text{def}}{=} \forall a \in V.\ [\texttt{pres}(a) \oplus \texttt{abs}(a)].$

$\texttt{well\_defined}(V) \stackrel{\text{def}}{=} \texttt{well\_defined}_0(V), \texttt{well\_defined}_1(V).$

– *The system*:

$\texttt{vars} \stackrel{\text{def}}{=} \{\texttt{p53}, \texttt{Mdm2}, \texttt{DNAdam}\}.$

$\texttt{rule}(1) \stackrel{\text{def}}{=} \texttt{inhib}(\texttt{vars}, \texttt{DNAdam}, \texttt{Mdm2}).$ $\texttt{rule}(4) \stackrel{\text{def}}{=} \texttt{inhib}(\texttt{vars}, \texttt{Mdm2}, \texttt{p53}).$

$\texttt{rule}(2) \stackrel{\text{def}}{=} \texttt{inhib}_\texttt{s}(\texttt{vars}, \texttt{Mdm2}, \texttt{p53}).$ $\texttt{rule}(5) \stackrel{\text{def}}{=} \texttt{inhib}_\texttt{c}(\texttt{vars}, \texttt{p53}, \texttt{DNAdam}).$

$\texttt{rule}(3) \stackrel{\text{def}}{=} \texttt{active}(\texttt{vars}, \texttt{p53}, \texttt{Mdm2}).$ $\texttt{rule}(6) \stackrel{\text{def}}{=} \texttt{inhib}_\texttt{s}(\texttt{vars}, \texttt{DNAdam}, \texttt{Mdm2}).$

$\texttt{system} \stackrel{\text{def}}{=} \texttt{vars}, \texttt{rule}(1), \texttt{rule}(2), \texttt{rule}(3),$
$\texttt{rule}(4), \texttt{rule}(5), \texttt{rule}(6), \texttt{well\_defined}(\texttt{vars}).$

– *Initial state*:

$\texttt{initial\_state} \stackrel{\text{def}}{=} \texttt{abs}(\texttt{p53}) \otimes \texttt{pres}(\texttt{Mdm2}), \quad \texttt{initial\_state}\ \texttt{at}\ 0.$

Fig. 2. Representation of the System in HyLL

4.3 Proofs

Although linear logic is well suited to describing transition systems, as we do here in the area of biology, this logic can sometimes be too precise in its resource management for our needs. To solve this constraint, we sometimes make precise that we do not care about the value of some variables. We define a `dont_care` predicate for this purpose:

$$\texttt{dont\_care}(x) \stackrel{\text{def}}{=} \texttt{pres}(x) \oplus \texttt{abs}(x) \qquad \texttt{dont\_care}(V) \stackrel{\text{def}}{=} \otimes_{x \in V} \texttt{dont\_care}(x).$$

This predicate is used in the statement of two of the four properties we present in this paper (Properties 1 and 4).

Additionally, in some proofs, when we do not know the value of some variables of the system, we sometimes need to perform a case analysis on their two possible values, using the $\texttt{well\_defined}_1$ predicate introduced for this purpose.

Finally let us give two definitions in order to further shorten propositions and proofs ($state_0$ is a state equivalent to the initial state):

$$state_0 \stackrel{\text{def}}{=} \texttt{abs}(\texttt{p53}) \otimes \texttt{pres}(\texttt{Mdm2}) \qquad state_1 \stackrel{\text{def}}{=} \texttt{pres}(\texttt{p53}) \otimes \texttt{abs}(\texttt{Mdm2}).$$

Property 1 As long as there is DNA damage, the above system can oscillate (with a short period) from $state_0$ to $state_1$ and back again. We outline the proof informally for this property only. Others are omitted. We refer the reader to the electronic appendix (for both the sequent proofs and their formalization).

From $state_0$ and $\texttt{pres}(\texttt{DNAdam})$ we get $\texttt{abs}(\texttt{p53})$, $\texttt{abs}(\texttt{Mdm2})$, and $\texttt{pres}(\texttt{DNAdam})$ by rule 1. Then $\texttt{pres}(\texttt{p53})$, $\texttt{abs}(\texttt{Mdm2})$, and $\texttt{pres}(\texttt{DNAdam})$ ($state_1$) by rule 2. Then $\texttt{pres}(\texttt{p53})$, $\texttt{pres}(\texttt{Mdm2})$, and $\texttt{pres}(\texttt{DNAdam})$ by rule 3, and finally $\texttt{abs}(\texttt{p53})$, $\texttt{pres}(\texttt{Mdm2})$, and $\texttt{pres}(\texttt{DNAdam})$ ($state_0$) by rule 4.

We define (and prove) our property in the two possible ways discussed earlier (Sect. 4.2), roughly corresponding to Definitions 7 and 9, respectively. The difference here is that our initial state (the one from which the oscillation starts) includes the presence of DNA damage.

Proposition (Property 1, Version 1). *For any world w, there exists two worlds u and v such that both u and v are less than 3 and the following holds:*

$$\dagger\, system \mathbin{@} 0 \;;\; state_0 \otimes \texttt{pres}(\texttt{DNAdam}) \mathbin{@} w \vdash \delta_u[(state_1 \otimes \texttt{dont\_care}(\texttt{DNAdam})) \;\&\; (\delta_v(state_0 \otimes \texttt{dont\_care}(\texttt{DNAdam})))] \mathbin{@} w$$

Alternatively, our property can be defined:

Proposition (Property 1, Version 2). *For any world w, there exists two worlds u and v such that both u and v are less than 3 and the following holds:*

$$\dagger\, system \mathbin{@} 0 \;;\; state_0 \otimes \texttt{pres}(\texttt{DNAdam}) \mathbin{@} w \vdash state_1 \otimes \texttt{dont\_care}(\texttt{DNAdam}) \mathbin{@} w.u \quad and$$
$$\dagger\, system \mathbin{@} 0 \;;\; state_1 \mathbin{@} w.u \vdash state_0 \mathbin{@} w.u.v$$

There are no $\texttt{dont\_care}$'s needed in the conclusion of the second sequent because only rules 3 and 4 are used, which don't involve DNAdam.

Property 2 DNA damage can be quickly recovered. This property can be stated directly as follows.

Proposition (Property 2). *For any world w, there exists a world u such that u is less than 5 and the following holds:*

$$\dagger\, system \mathbin{@} 0;\; state_0 \otimes \texttt{pres}(\texttt{DNAdam}) \mathbin{@} w \vdash state_0 \otimes \texttt{abs}(\texttt{DNAdam}) \mathbin{@} w.u$$

Induction/Case Analysis Most of interesting proofs require case analysis or induction; this is the case for Properties 3 and 4 below. More precisely, we need here *case analysis on the set of fireable rules*. We implement this by a case analysis on the interval [1..6] of our six rules, together with a new predicate `fireable` defining the necessary conditions for each rule to fire. We shall also need the negation of this predicate: `not_fireable`. We give here the definitions of these predicates for the first rule of our system, for both (strong and general) styles of the rules used in the present paper (the complete definitions can be found in Appendix B):

$$\begin{array}{l}
\texttt{fireable}_{\texttt{s}}(1) \stackrel{\text{def}}{=} \texttt{pres}(\texttt{DNAdam}) \otimes \texttt{pres}(\texttt{Mdm2}) \otimes \texttt{dont_care}(\texttt{p53}) \\
\texttt{not_fireable}_{\texttt{s}}(1) \stackrel{\text{def}}{=} \\
\quad ((\texttt{abs}(\texttt{DNAdam}) \otimes \texttt{pres}(\texttt{Mdm2})) \ \oplus \ (\texttt{pres}(\texttt{DNAdam}) \otimes \texttt{abs}(\texttt{Mdm2})) \ \oplus \\
\quad (\texttt{abs}(\texttt{DNAdam}) \otimes \texttt{abs}(\texttt{Mdm2}))) \ \otimes \ \texttt{dont_care}(\texttt{p53}) \\
\texttt{fireable}(1) \stackrel{\text{def}}{=} \\
\quad (\texttt{pres}(\texttt{DNAdam}) \oplus (\texttt{pres}(\texttt{DNAdam}) \otimes \texttt{pres}(\texttt{Mdm2})) \oplus \\
\quad (\texttt{pres}(\texttt{DNAdam}) \otimes \texttt{abs}(\texttt{Mdm2}))) \ \otimes \ \texttt{dont_care}(\texttt{p53}) \\
\texttt{not_fireable}(1) \stackrel{\text{def}}{=} \texttt{abs}(\texttt{DNAdam}) \ \otimes \ \texttt{dont_care}(\{\texttt{Mdm2}, \texttt{p53}\})
\end{array}$$

An (informal) formula like "for any *fireable rule* r, P" will be written as "for any rule r in the interval [1..6], the following holds: $(\texttt{fireable}(r) \ \& \ P) \oplus \texttt{not\_fireable}(r)$". Note that both definitions of `fireable` and `not_fireable` could be generated from the definitions of the biological rules, as the rules we consider in the present paper always have the same shape.

Property 3 If there is no DNA damage, the system remains in the initial state. A first attempt at formalizing this property might be:

For any world w, the following holds: $\dagger\, system \ @ \ 0, \ \texttt{abs}(\texttt{DNAdam}) \ @ \ 0 \vdash state_0 \otimes \texttt{abs}(\texttt{DNAdam}) \ @ \ w$.

However, the above statement does not model our property. We want to prove that if $\texttt{abs}(\texttt{DNAdam}) \ @ \ 0$ then $state_0 \otimes \texttt{abs}(\texttt{DNAdam}) \ @ \ w$ holds, for all worlds w, *no matter which rule is fired* to get to w. Thus our property requires a *case analysis* on the rules of the biological system.

Proposition (Property 3). *Let $\mathcal{P}$ denote the formula $state_0 \otimes \texttt{abs}(\texttt{DNAdam})$. For any world w, the following holds: $\dagger\, system \ @ \ 0, \ \mathcal{P} \ @ \ 0 \ \vdash \mathcal{P} \ \texttt{at} \ 0 \ @ \ w$; and for any world w, for any rule r in the interval* [1..6]*, the following holds:*

$$\dagger\, system \ @ \ 0 \vdash \mathcal{P} \rightarrow (\texttt{fireable}(r) \ \& \ \delta_1 \, \mathcal{P}) \ \oplus \ \texttt{not\_fireable}(r) \ @ \ w$$

The proof of the second statement proceeds by case analysis on the rules (r) of the biological system. There are only two fireable rules: $\texttt{rule}(4)$ and $\texttt{rule}(6)$.

Property 4 There is no path with two consecutive states where p53 and Mdm2 are both present or both absent. In other words: from any state where p53 and Mdm2 are both present or both absent, we can only go to a state where either p53 is present and Mdm2 is absent or p53 is absent and Mdm2 is present.

This requires a stronger (natural) hypothesis: we need the property that each rule modifies at least one entity in the system. In order to achieve this, we shall use the strong style of definitions for our inhibition and activation rules discussed earlier (Sect. 4.2). For example, the activation rule will be defined as follows:

$$s\_\mathtt{active}(V, a, b) \stackrel{\text{def}}{=} \mathtt{pres}(a) \otimes \mathtt{abs}(b) \rightarrow \\ \delta_1(\mathtt{pres}(a) \otimes \mathtt{pres}(b)) \otimes \downarrow u.\ \mathtt{unchanged}(V \setminus \{a, b\}, u)).$$

The complete set of strong rules can be found in Appendix B.

Let $\mathcal{L}$ and $\mathcal{R}$ denote the following two formulas:

$$\mathcal{L} := (\mathtt{pres(p53)} \otimes \mathtt{pres(Mdm2)}) \oplus (\mathtt{abs(p53)} \otimes \mathtt{abs(Mdm2)})$$
$$\mathcal{R} := ((\mathtt{pres(p53)} \otimes \mathtt{abs(Mdm2)}) \oplus \\ (\mathtt{abs(p53)} \otimes \mathtt{pres(Mdm2)})) \otimes \mathtt{dont\_care(DNAdam)}$$

We want to prove that from state $\mathcal{L}$ we can only go to state $\mathcal{R}$, *no matter which rule is fired*. Here again, we need *case analysis on the set of fireable rules*:

Proposition (Property 4). *For any world w, for any rule r in the interval* [1..6]*, the following holds:*

$\dagger\, system\ @\ 0;\ . \vdash \mathcal{L} \rightarrow (s\_\mathtt{fireable}(r)\ \&\ \delta_1\, \mathcal{R}) \oplus s\_\mathtt{not\_fireable}(r)\ @\ w$

Property 4 could be written as the CTL formula $\mathsf{AG}(\mathcal{L} \rightarrow \mathsf{AX}\mathcal{R})$ (see Sect. 6 for the encoding of such a formula in HyLL). Nevertheless, to simplify the proof we can observe that the argument property of AG contains an implication and thus all the possible states verifying the left hand side of the implication should be taken into account in the proof. As a matter of fact, at each step we do not make assumptions on the state where $\mathcal{L}$ holds. The system satisfies Property 4 if all its states satisfy $\mathcal{L} \rightarrow \mathsf{AX}\mathcal{R}$. Since in our transition system all the states are reachable from the initial states, this corresponds to requiring the satisfaction of Property 4 at (that is, from) the initial states. In case we want to test such a property at a state S_i that is not connected to all the other states, we only need to prove the property in the subtree of the transition system rooted in S_i. In HyLL, we should prove the following theorem: $if\ reachable(S, S_i)\ then\ S \vdash \mathcal{L} \rightarrow \forall r \in R.\ \delta_1 \mathcal{R}$, where $reachable(S, S_i) \stackrel{\text{def}}{=} \exists u.\ S_i \rightarrow \delta_u S$.

5 Formal Proofs

Our approach is to fully formalize proofs in Coq, using the λProlog prover to help with partial automation of the proofs. The λProlog prover is a tactic-style interactive theorem prover implemented in the manner described in [16], also given as an example in Sect. 9.4 of [27]. The code described there can be viewed as a logical framework, implementing a tactic-style architecture. We use this code directly, and instantiate the framework with tactics implementing the basic inference rules of HyLL.

We use Coq for two reasons. First, we can build libraries in Coq that allow us to reason at two levels. We can prove meta-level properties of HyLL (for example, we have formalized Theorem 2), and we can reason at the object-level, which in this case means that we can prove HyLL sequents directly. To do so, we adopt the two-level style of reasoning used in Hybrid [17] where the logic we want to reason in and about is called the *specification logic* and is implemented as an inductive predicate in Coq. The inductive predicate in this case defines HyLL sequents, and its definition is a fairly direct modification of the ordered linear logic in [17]. Second, once a proof is complete, Coq provides a *proof certificate*. In particular, Coq implements the calculus of inductive contstructions (CIC), where a property is stated as a type in CIC and a proof is a λ-term inhabiting that type. This λ-term serves as a certificate, which can be stored and checked independently.

It is, of course, possible to prove the properties in Sect. 4 only using Coq, but in general, proofs in Coq of HyLL sequents quickly become cumbersome because of the amount of detail required to apply each inference rule of HyLL. The λProlog prover is used to automatically infer much of this detail. For example, because we implement HyLL directly as an inductive predicate in Coq, in order to apply the $\otimes L$ rule (see Fig. 1), Coq's `apply` tactic requires arguments to be given explicitly for the instantiation of formulas A and B, world u, and multisets Δ and $\Delta, A \mathbin{@} u, B \mathbin{@} u$. (Using the more flexible `eapply` does not help with the proofs considered here.) As a result, the Coq proofs are verbose and often contain redundant information. In λProlog on the other hand, our primitive tactics for applying HyLL inference rules use unification to infer these arguments. We could instead program a more automated version of the `apply` tactic in Coq, tailored to applying HyLL rules using Coq's Ltac facility, but this task would likely be more complex and adhoc, since unification is not one of the primitive operations of Coq.

It is also straightforward to represent HyLL proof terms in λProlog, and implement the construction of these proof terms as part of the implementation of the primitive tactics. In our λProlog prover, we construct such proof terms and then translate them to strings representing Coq proof script. This translation is also implemented in λProlog. These automatically generated proof scripts are imported into Coq, and then after some fine-tuning of the script, we obtain a proof certificate for the entire proof.

Note that we do not use λProlog to fully automate proofs; the λProlog prover is also interactive. The user indicates what HyLL rule to apply at each step, possibly with some other basic information such as the position in Δ of the formula to which the rule is to be applied. The arguments needed to apply the rule are then inferred automatically. We could build more automation into the prover, possibly in the style of the linear logic programming language presented in [21] for general automation, with the addition of heuristics tailored to our application. This is left for future work.

The reader is referred to the electronic appendix for the encoding of our biological system in Coq. Here, for illustration purposes, we simply state and discuss Property 3 exactly as it appears in Coq.

```
Theorem Property3 : forall w:world,
  seq Gamma ((PP @ 0)::nil) ((PP at 0) @ w) /\
  forall (n:nat) (A B:oo_), fireable n A -> not_fireable n B ->
  seq Gamma nil ((PP ->> ((A &a step PP) +o B)) @ w).
```

In general, sequents have the form `(seq Gamma Delta (A @ W))`, where `A` represents a HyLL formula (of type `oo_` in the Coq encoding) and `W` is a world, which are encoded using Coq's built-in type for natural numbers. `Gamma` is a list of HyLL formulas (using the built-in lists of Coq) and `Delta` is a multiset of elements of type `oo_`, where we build our own custom multiset library. In the above theorem, `Gamma` is the Coq encoding of $(\dagger\, system \,@\, 0)$. In the first sequent in the statement of the theorem, `Delta` contains only `(PP @ 0)` where `PP` is the Coq encoding of $(state_0 \otimes \texttt{abs(DNAdam)})$, and in the second sequent `Delta` is empty. The symbols and constants `@`, `at`, `->>`, `&a`, `step`, and `+o` represent the HyLL operators @, `at`, $\rightarrow$, &, δ_1, and $\oplus$, respectively. The predicates `fireable` and `not_fireable` are defined inductively in Coq, where the first argument is a natural number specifying the rule number.

The Coq proof of the second conjunct proceeds by case analysis on `n`, followed by inversion on `(fireable n A)` and `(not_fireable n B)`, which provides instantiations for `A` and `B` (the conditions that express whether the rule is fireable or not). The resulting 6 subgoals are sent to the λProlog prover, whose output is imported back into Coq as described above.

6 Comparison with Model Checking

While temporal logics such as LTL, CTL, or CTL* have been very successful in practice with efficient model checking tools, the proof theory of these logics is very complex. In contrast, HyLL has a very traditional proof theoretic pedigree: it is presented in the sequent calculus and enjoys cut-elimination and focusing [8]. A further advantage of our approach with respect to model checking is that it provides an unified framework to encode both transition rules and (both statements and proofs of) temporal properties.

Let us examine both approaches in more details.

6.1 Temporal Operators

We propose the following encoding of temporal logic operators in HyLL$[\mathcal{T}]$, where $\mathcal{T} = \langle I\!N, +, 0\rangle$, representing instants of time. While this domain does not have any branching structure like CTL, it is expressive enough for many common idioms because of the branching structure of derivations involving $\oplus$.

State quantifiers can be easily mapped. There is a clear correspondence between F (resp. G) and $\Diamond$ (resp. $\Box$), see Definition 6. The encodings of X and U are the following ones: $\mathsf{X}P \Leftrightarrow \delta_1 P$ and $P_1 \mathsf{U} P_2 \Leftrightarrow\!\downarrow u.\ \exists v.\ P_2\ \texttt{at}\ u.v \otimes \forall w < v.\ P_1\ \texttt{at}\ u.w.$ where P, P_1, and P_2 are some propositions (not necessarily atomic ones).[4]

[4] Observe that a proposition characterizes a set of states.

As for path quantifiers, the question is more subtle. The idea is that E corresponds to the existence of a proof, while for A it is necessary to look at a proof considering all the possible rules to be applied at each step (at each step of the proof, the chosen rule should not influence the property satisfaction).

We came to the conclusion that the encoding of A in HyLL depends on the state quantifier following it. Let R be the set of rules of our transition system. The mapping we propose is the following one:

- $\mathsf{AX}P$. In HyLL, we write $\forall r \in R\ \delta_1 P$. More precisely, the encoding contemplating fireable rules is $\forall r \in R\ (\texttt{fireable}(r)\ \&\ \delta_1 P) \oplus \texttt{not\_fireable}(r)$ (see Sect. 4.3). For the sake of simplicity, in the following we omit such details concerning fireable rules.
- $\mathsf{AG}P$. It is equivalent to $P \wedge \mathsf{AG}(P \rightarrow \mathsf{AX}(P))$. In HyLL, we write $P \otimes \forall n(P\ at\ n) \rightarrow \forall r \in R(P \text{ at } n+1)$.
- $\mathsf{AF}P$. It is equivalent to $P \vee \mathsf{AX}(\mathsf{AF}P)$. If we have a bound k on the number of steps needed to satisfy the property, we can expand this formula by obtaining: $P \vee \mathsf{AX}(P \vee \mathsf{AX}(\ldots \mathsf{AX}P))$, with k nested occurrences of AX. In HyLL, we write $P \oplus \forall r \in R(\delta_1 P \oplus (\forall r \in R(\ldots \delta_k P)))$. Notice that another alternative is to express the F operator by using U ($\mathsf{F}P \Leftrightarrow \text{true}\ \mathsf{U}P$).
- $\mathsf{A}(P_1 \mathsf{U} P_2)$. It is equivalent to $P_2 \vee (P_1 \wedge \mathsf{AX}(P_1 \mathsf{U} P_2)$. If we have a bound k on the number of steps needed to satisfy the property, we can expand this formula by obtaining: $P_2 \vee (P_1 \wedge \mathsf{AX}(P_2 \vee (P_1 \wedge \mathsf{AX}(\ldots \mathsf{AX}P_2))))$, with k nested occurrences of AX. In HyLL, we write $P_2 \oplus (P_1 \otimes \forall r \in R(\delta_1 P_2 \oplus (\delta_1 P_1 \otimes \forall r \in R(\ldots \delta_k P_2))))$.

In addition to the future connectives, the domain $\mathcal{T}$ also admits past connectives if we add saturating subtraction (*i. e.*, $a - b = 0$ if $b \geq a$) to the language of worlds. We can then define the duals H (historically) and O (once) of G and F as:

$$\mathsf{H}\ P \stackrel{\text{def}}{=}\downarrow u.\forall w.(P\ \texttt{at}\ u - w) \qquad \mathsf{O}\ P \stackrel{\text{def}}{=}\downarrow u.\exists w.(P\ \texttt{at}\ u - w)$$

6.2 Model Checking

A strength of our approach with respect to model checking is that, when we prove an existential property using certain rules of a model, we have the guarantee that *all* the models containing such rules satisfy the property. This is important because in biology we often deal with incomplete information. It is also worth noting that in model checking, all objects are finite: both the number of states, and the number of transitions in the state graph. In HyLL, objects can potentially be infinite; in particular, we can have an infinite number of states. Let us point out further advantages of our approach with respect to model checking. First of all, when proving a given property we do not need to blindly try all possible rules at each step but we can guide the proof (see Sect. 7). Observe that a successful proof of a given property can be exploited to prove similar properties. Furthermore, suppose we are able to prove a property of the system which is not desirable. In this case the proof we get can help us in understanding what should

be modified in the system so that the property is not satisfied. More precisely, we can look for the rules to be removed/modified among those that have been used in the proof. In model checking, when a property turns out to be true, the reason is not investigated. Finally, in [34], temporal logic is extended to allow the expression of properties such as "P is true at every even state of an infinite path." A decision procedure for this extended logic is also defined, but to the best of our knowledge, there is no model checking tool for it. In HyLL, if we add equality on worlds, we can write $\forall n = 2k.\ P\ at\ n$.

Note that in some of the temporal properties we test, there is a bound on the number of time units, and thus on the length of the proof. In this particular case, there is a strong analogy with "bounded model checking" [3].

A drawback of theorem proving with respect to model checking is that this method can be time consuming and needs an expert. Recent advances in both proof theory and systems however provide us with at least partial, and sometimes complete, automation of the proofs.

7 Conclusion and Future Work

In this paper we argued that the HyLL logic can be successfully exploited for formally verifying Boolean biological systems. This work is a first experiment along this new line of research (although we already provide fully mechanized proofs). We focussed on a simple regulatory network but our framework could be adopted to model several other kinds of biological networks (e.g., neuronal, predator-prey, or ecological networks).

A natural extension of this work consists of applying our methodology to multivalued, continuous, and stochastic biological models. As far as the first case is concerned, the extension is straightforward, we just need to replace present/absent predicates by predicates indicating the discrete values of variables. The latter two cases are more involved. We could try to use the domain of world $\mathcal{W} = \langle \Re^+, +, 0 \rangle$, use predicates to represent variable concentrations and express the evolution of each variable in terms of functions involving kinetic expressions such as the mass action law, Hill, or Michaelis-Menten kinetics. In [8], several alternatives for the worlds in the probabilistic case are also discussed. In any case, the challenge would consist of being able to perform symbolic calculation as much as possible without evaluating functions.

Proofs of properties such as 1 and 2 require finding a path through the system, which here means specifying a series of rules that can be applied in a particular order. At each step, there may be a choice between several potential fireable rules. In the interactive proofs, such choices were made by hand. Our future work includes building automated procedures (e.g., Coq tactics) to guide the proof. We would also like to extend our model to include axioms for events such as those considered in Biocham, which make it possible to change the value of some variables under certain special conditions. Such events often correspond to external inputs and have priority over the ordinary rules of a model.

Our aim is to find the logical essence of biochemical reactions. What we envision for the domain of "biological computation" is a resource-aware stochastic or probabilistic λ-calculus that has HyLL propositions as (behavioral) types.

References

1. Alur, R., Belta, C., Ivaničić, F., Kumar, V., Mintz, M., Pappas, G.J., Rubin, H., Schug, J.: Hybrid modeling and simulation of biomolecular networks. In: Fourth International Workshop on Hybrid Systems: Computation and Control. Springer LNCS, vol. 2034, pp. 19–32 (2001)
2. Bertot, Y., Castéran, P.: Interactive Theorem Proving and Program Development. Coq'Art: The Calculus of Inductive Constructions. Springer (2004)
3. Biere, A., Cimatti, A., Clarke, E.M., Strichman, O., Zhu, Y.: Bounded model checking. In: Zelkowitz, M. (ed.) Highly Dependable Software, pp. 117–148. No. 58 in Advances in Computers, Elsevier (2003)
4. Bozzano, M.: A Logic-Based Approach to Model Checking of Parameterized and Infinite-State Systems. Ph.D. thesis, DISI, Università di Genova (2002)
5. Campagna, D., Piazza, C.: Hybrid automata in systems biology: How far can we go? Electronic Notes in Theoretical Computer Science 229(1), 93–108 (2009)
6. Cervesato, I., Pfenning, F., Walker, D., Watkins, K.: A concurrent logical framework II: Examples and applications. Tech. Rep. CMU-CS-02-102, Carnegie Mellon University (2003)
7. Chabrier-Rivier, N., Chiaverini, M., Danos, V., Fages, F., Schächter, V.: Modeling and querying biochemical interaction networks. Theoretical Computer Science 325(1), 25–44 (Sep 2004)
8. Chaudhuri, K., Despeyroux, J.: A hybrid linear logic for constrained transition systems with applications to molecular biology. Tech. Rep. inria-00402942, INRIA-HAL (October 2013)
9. Ciliberto, A., Novák, B., Tyson, J.J.: Steady states and oscillations in the p53/Mdm2 network. Cell Cycle 4(3), 488–493 (Mar 2005)
10. Cimatti, A., Clarke, E.M., Giunchiglia, F., Roveri, M.: NuSMV: A new symbolic model verifier. In: Eleventh International Conference on Computer Aided Verification. Springer LNCS, vol. 1633, pp. 495–499 (1999)
11. Clarke, Jr., E.M., Grumberg, O., Peled, D.A.: Model checking. MIT Press, Cambridge, MA, USA (1999)
12. Courcoubetis, C., Vardi, M.Y., Wolper, P., Yannakakis, M.: Memory-efficient algorithms for the verification of temporal properties. Formal Methods in System Design 1(2/3), 275–288 (1992)
13. Danos, V., Laneve, C.: Formal molecular biology. Theoretical Computer Science 325(1), 69–110 (Sep 2004)
14. Despeyroux, J., Chaudhuri, K.: A hybrid linear logic for constrained transition systems (2014), to appear in Types for Proofs and Programs, post-proceedings of TYPES 2013, LIPIcs (Leibniz International Proceedings in Informatics)
15. Fages, F., Soliman, S., Chabrier-Rivier, N.: Modelling and querying interaction networks in the biochemical abstract machine BIOCHAM. Journal of Biological Physics and Chemistry 4(2), 64–73 (2004)
16. Felty, A.: Implementing tactics and tacticals in a higher-order logic programming language. Journal of Automated Reasoning 11(1), 43–81 (Aug 1993)

17. Felty, A.P., Momigliano, A.: Hybrid: A definitional two-level approach to reasoning with higher-order abstract syntax. Journal of Automated Reasoning 48(1), 43–105 (2012)
18. Gentzen, G.: Investigations into logical deductions, 1935. In: Szabo, M.E. (ed.) The Collected Papers of Gerhard Gentzen, pp. 68–131. North-Holland Publishing Co., Amsterdam (1969)
19. Girard, J.Y.: Linear logic. Theoretical Computer Science 50, 1–102 (1987)
20. Hinton, A., Kwiatkowska, M., Norman, G., Parker, D.: PRISM: A tool for automatic verification of probabilistic systems. In: Twelfth International Conference on Tools and Algorithms for the Construction and Analysis of Systems. Springer LNCS, vol. 3920, pp. 441–444 (2006)
21. Hodas, J.S., Miller, D.: Logic programming in a fragment of intuitionistic linear logic. Journal of Information and Computation 110(2), 327–365 (1994)
22. Hofestädt, R., Thelen, S.: Quantitative modeling of biochemical networks. In: In Silico Biology, vol. 1, pp. 39–53. IOS Press (1998)
23. Holzmann, G.J.: The Spin Model Checker: Primer and Reference Manual. Addison-Wesley Professional (2003)
24. de Jong, H., Gouzé, J.L., Hernandez, C., Page, M., Sari, T., Geiselmann, J.: Qualitative simulation of genetic regulatory networks using piecewise-linear models. Bulletin of Mathematical Biology 66(2), 301–340 (2004)
25. Maria, E.D., Fages, F., Rizk, A., Soliman, S.: Design, optimization and predictions of a coupled model of the cell cycle, circadian clock, DNA repair system, irinotecan metabolism and exposure control under temporal logic constraints. Theoretical Computer Science 412(21), 2108–2127 (2011)
26. Miller, D.: The π-calculus as a theory in linear logic: Preliminary results. In: 3rd Workshop on Extensions to Logic Programming. Springer LNCS, vol. 660, pp. 242–265 (1993)
27. Miller, D., Nadathur, G.: Programming with Higher-Order Logic. Cambridge University Press (2012)
28. Phillips, A., Cardelli, L.: A correct abstract machine for the stochastic pi-calculus. In: BioConcur: Workshop on Concurrent Models in Molecular Biology. Electronic Notes in Theoretical Computer Science (2004)
29. Reddy, V.N., Mavrovouniotis, M.L., Liebman, M.N.: Petri net representations in metabolic pathways. In: First International Conference on Intelligent Systems for Molecular Biology. pp. 328–336. AAAI Press (1993)
30. Regev, A., Panina, E.M., Silverman, W., Cardelli, L., Shapiro, E.: Bioambients: An abstraction for biological compartments. Theoretical Computer Science 325(1), 141–167 (Sep 2004)
31. Regev, A., Silverman, W., Shapiro, E.Y.: Representation and simulation of biochemical processes using the π-calculus process algebra. In: Sixth Pacific Symposium on Biocomputing. pp. 459–470 (2001)
32. Thomas, R.: Boolean formalization of genetic control circuits. Journal of Theoretical Biology 42(3), 563–585 (Dec 1973)
33. Thomas, R., Thieffry, D., Kaufman, M.: Dynamical behaviour of biological regulatory networks—I. Biological role of feedback loops and practical use of the concept of the loop-characteristic state. Bulletin of Mathematical Biology 57(2), 247–276 (1995)
34. Wolper, P.: Temporal logic can be more expressive. Information and Control 56(1/2), 72–99 (1983)