Computation Tree Regular Logic for Genetic Regulatory Networks

Radu Mateescu[1], Pedro T. Monteiro[1,2], Estelle Dumas[1], and Hidde de Jong[1]

[1] INRIA Rhône-Alpes, 655 Av. de l'Europe, F-38330 Montbonnot St Martin, France
[2] INESC-ID/IST, Rua Alves Redol 9, 1000-029 Lisboa, Portugal
{Radu.Mateescu,Pedro.Monteiro,Estelle.Dumas,Hidde.de-Jong}@inrialpes.fr

Abstract. Model checking has proven to be a useful analysis technique not only for concurrent systems, but also for the genetic regulatory networks (GRNS) that govern the functioning of living cells. The applications of model checking in systems biology have revealed that temporal logics should be able to capture both branching-time and fairness properties. At the same time, they should have a user-friendly syntax easy to employ by non-experts. In this paper, we define CTRL (Computation Tree Regular Logic), an extension of CTL with regular expressions and fairness operators that attempts to match these criteria. CTRL subsumes both CTL and LTL, and has a reduced set of temporal operators indexed by regular expressions, inspired from the modalities of PDL (Propositional Dynamic Logic). We also develop a translation of CTRL into HMLR (Hennessy-Milner Logic with Recursion), an equational variant of the modal μ-calculus. This has allowed us to obtain an on-the-fly model checker with diagnostic for CTRL by directly reusing the verification technology available in the CADP toolbox. We illustrate the application of the CTRL model checker by analyzing the GRN controlling the carbon starvation response of *Escherichia coli.*

1 Introduction

Explicit state verification has been mostly applied to the analysis of concurrent systems in engineering. Recently, however, biological regulatory networks have been recognized as special cases of concurrent systems as well, which has opened the way for the application of formal verification technology in the emerging field of systems biology (see [1,2] for reviews). The networks controlling cellular functions consist of genes, proteins, small molecules, and their mutual interactions. Most of these networks are large and complex, thus defying our capacity to understand how the dynamic behavior of the cell emerges from the structure of interactions. A large number of mathematical formalisms have been proposed to describe these networks [3], giving rise to models that can be directly or indirectly mapped to Kripke structures.

The representation of the dynamics of biological regulatory networks by means of Kripke structures enables the application of formal verification techniques to the analysis of properties of the networks, formulated as queries in temporal logic.

Several applications of model checking exists in the bioinformatics and systems biology literature [4,5,6,7,8,9,10]. In our previous work [11,6], we have developed GNA (*Genetic Network Analyzer*), a tool for the qualitative simulation of genetic regulatory networks, and connected it to state-of-the-art model checkers like NuSMV [12] and CADP [13].

The application to actual biological systems brought a few properties of the network dynamics to the fore that are not easily expressed in classical temporal logics. For instance, questions about multistability are important in the analysis of biological regulatory networks [14], but difficult (or impossible) to express in LTL [15]. CTL [16] is capable of dealing with branching time, important for multistability and other properties of non-deterministic models. However, it is not expressive enough to specify the occurrence of oscillations of indefinite length, a special kind of fairness property [6]. An obvious solution would be to consider CTL* [17] or the propositional μ-calculus [18], both of which subsume CTL and LTL; however, these powerful branching-time logics are complex to understand and use by non-experts. More generally, it is not easy to express *observations* in temporal logic. Often these take the form of patterns of events corresponding to variations of system variables (protein concentrations, their derivatives, etc.) measured by experiments in the lab, which can be compared with the model predictions and thus help validate the model. Observations are conveniently and concisely formulated in terms of regular expressions, but these are not provided by standard temporal logics such as CTL and LTL.

In this paper, we aim at providing a temporal specification language that allows expressing properties of biological interest and strikes a suitable compromise between expressive power, user-friendliness, and complexity of model checking. Towards this objective, we propose a specification language named CTRL (*Computation Tree Regular Logic*), which extends CTL with regular expressions and fairness operators. CTRL is more expressive than previous extensions of CTL with regular expressions, such as RCTL [19] and RegCTL [20], whilst having a simpler syntax due to a different choice of primitive temporal operators, inspired from dynamic logics like PDL [21]. CTRL also subsumes CTL, LTL, and PDL-Δ [22] allowing in particular the concise expression of bistability and oscillation properties. Although CTRL was primarily designed for describing properties of regulatory networks in system biology, it also enables a succinct formulation of typical safety, liveness, and fairness properties useful for the verification of concurrent systems in other domains.

As regards the evaluation of CTRL formulas on Kripke structures, we adopt as verification engine CADP [13], a state-of-the-art verification toolbox for concurrent asynchronous systems that provides, among other functionalities, on-the-fly model checking and diagnostic generation for μ-calculus formulas on labeled transition systems (LTSs). In order to reuse this technology, we have to move from the state-based setting (CTRL and Kripke structures) to the action-based setting (μ-calculus and LTSs). The translation from Kripke structures to LTSs is done in the standard way [16]. The translation from CTRL to an action-based logic is carried out by considering as target language HMLR (HML *with recursion*) [23].

The equational representation of HMLR is closer to the boolean equation systems (BESs) used as intermediate formalism by the verification engine, namely the CÆSAR_SOLVE [24] generic library for local BES resolution.

The CTRL model checking procedure obtained in this way has a linear-time complexity w.r.t. the size of the formula and the Kripke structure for a significant part of the logic. This part notably subsumes PDL-Δ and allows the multistability and oscillation properties to be captured. The inevitability operator of CTRL and its infinitary version (inevitable looping) has an exponential worst-case complexity w.r.t. the size of its regular subformula; this complexity becomes linear, however, when the regular subformula is "deterministic" in a way similar to finite automata. In practice, the usage of CTRL and the model checker reveals that properties of biological interest can be expressed and verified efficiently. We illustrate this on the analysis of a model of the genetic regulatory network (GRN) controlling the carbon starvation response of *E. coli.*

The paper is organized as follows. Section 2 defines the syntax and semantics of CTRL and Section 3 presents the on-the-fly model checking procedure. Section 4 discusses the implementation of the CTRL model checker and applies it to the example of *E. coli.* Section 5 summarizes the results and provides directions for future work. A more extensive description of CTRL, including formal definitions and proofs, is available in [25].

2 Syntax and Semantics of CTRL

CTRL is interpreted on Kripke structures (KSs), which provide a natural formal description of concurrent systems, including biological regulatory networks. A KS is a tuple $K = \langle S, P, L, T, s_0 \rangle$, where: S is the set of states; P is a set of atomic propositions (predicates over states); $L : S \to 2^P$ is the state labeling (each state s is associated with the atomic propositions satisfied by s); $T \subseteq S \times S$ is the transition relation; and $s_0 \in S$ is the initial state. Transitions $(s_1, s_2) \in T$ are also noted $s_1 \to_T s_2$ (the subscript $_T$ is omitted if it is clear from the context). The transition relation T is assumed to be total, i.e., for each state $s_1 \in S$, there exists a transition $s_1 \to_T s_2$. A path $\pi = s_0 s_1 \ldots s_k \ldots$ is an infinite sequence of states such that $s_i \to_T s_{i+1}$ for every $i \geq 0$. The i-th state of a path π is noted π_i. The interval going from the i-th state of a path π to the j-th state of π inclusively (where $i \leq j$) is noted $\pi_{i,j}$. An interval $\pi_{0,i}$ is called prefix of π. For each state $s \in S$, $Path(s)$ denotes the set of all paths going out of s, i.e., the paths π such that $\pi_0 = s$. In the sequel, we assume the existence of a KS $K = \langle S, P, L, T, s_0 \rangle$, on which all formulas will be interpreted.

The syntax and semantics of CTRL are defined in the figure below. The logic contains state formulas φ and regular formulas ρ, which characterize properties of states and intervals, respectively. State formulas are built from atomic propositions $p \in P$ by using standard boolean operators and the EF, AF, EF^∞, AF^∞ temporal operators indexed by regular formulas ρ. Regular formulas are built from state formulas by using standard regular expression operators.

The interpretation $[\![\varphi]\!]_K$ of a state formula denotes the set of states of K that satisfy φ. The interpretation of regular formulas is defined by the satisfaction relation $\models_K$, which indicates whether an interval $\pi_{i,j}$ of a path in K satisfies a regular formula ρ (notation $\pi_{i,j} \models_K \rho$). The notation ρ^j (where $j \geq 0$) stands for the concatenation $\rho \ldots \rho$, where ρ occurs j times. The semantics of boolean operators is defined in the standard way. A state satisfies the potentiality formula $\mathsf{EF}_\rho\varphi$ iff it has an outgoing path containing a prefix satisfying ρ and leading to a state satisfying φ. A state satisfies the inevitability formula $\mathsf{AF}_\rho\varphi$ iff all of its outgoing paths contain a prefix satisfying ρ and lead to a state satisfying φ. A state satisfies the potential looping formula EF_ρ^∞ iff it has an outgoing path consisting of an infinite concatenation of intervals satisfying ρ. A state satisfies the inevitable looping formula AF_ρ^∞ iff all of its outgoing paths consist of an infinite concatenation of intervals satisfying ρ. An interval satisfies the one-step interval formula φ iff it consists of two states, the first of which satisfies φ. An interval satisfies the concatenation formula $\rho_1.\rho_2$ if it is the concatenation of two subintervals, the first one satisfying ρ_1 and the second one satisfying ρ_2. An interval satisfies the choice formula $\rho_1|\rho_2$ iff it satisfies either ρ_1, or ρ_2. An interval satisfies the iteration formula ρ^* iff it is the concatenation of (0 or more) subintervals satisfying ρ. By definition, an empty interval $\pi_{i,i}$ satisfies ρ^0 for any regular formula ρ. K satisfies φ (notation $K \models \varphi$) iff $s_0 \in [\![\varphi]\!]_K$.

Syntax

State formulas:

$$\varphi ::= p \mid \neg\varphi \mid \varphi_1 \vee \varphi_2 \mid \mathsf{EF}_\rho\varphi \mid \mathsf{AF}_\rho\varphi \mid \mathsf{EF}_\rho^\infty \mid \mathsf{AF}_\rho^\infty$$

Regular formulas:

$$\rho ::= \varphi \mid \rho_1.\rho_2 \mid \rho_1|\rho_2 \mid \rho^*$$

Semantics

State formulas:

$$\begin{aligned}
[\![p]\!]_K &= \{s \in S \mid p \in L(s)\}\\
[\![\neg\varphi]\!]_K &= S \setminus [\![\varphi]\!]_K\\
[\![\varphi_1 \vee \varphi_2]\!]_K &= [\![\varphi_1]\!]_K \cup [\![\varphi_2]\!]_K\\
[\![\mathsf{EF}_\rho\varphi]\!]_K &= \{s \in S \mid \exists\pi \in \mathit{Path}_K(s).\exists i \geq 0.\pi_{0,i} \models_K \rho \wedge \pi_i \in [\![\varphi]\!]_K\}\\
[\![\mathsf{AF}_\rho\varphi]\!]_K &= \{s \in S \mid \forall\pi \in \mathit{Path}_K(s).\exists i \geq 0.\pi_{0,i} \models_K \rho \wedge \pi_i \in [\![\varphi]\!]_K\}\\
[\![\mathsf{EF}_\rho^\infty]\!]_K &= \{s \in S \mid \exists\pi \in \mathit{Path}_K(s).\forall j \geq 0.\exists i \geq 0.\pi_{0,i} \models_K \rho^j\}\\
[\![\mathsf{AF}_\rho^\infty]\!]_K &= \{s \in S \mid \forall\pi \in \mathit{Path}_K(s).\forall j \geq 0.\exists i \geq 0.\pi_{0,i} \models_K \rho^j\}
\end{aligned}$$

Regular formulas:

$$\begin{aligned}
&\pi_{i,j} \models_K \varphi && \text{iff } j = i + 1 \wedge \pi_i \in [\![\varphi]\!]_K\\
&\pi_{i,j} \models_K \rho_1.\rho_2 && \text{iff } \exists k \in [i,j].\pi_{i,k} \models_K \rho_1 \wedge \pi_{k,j} \models_K \rho_2\\
&\pi_{i,j} \models_K \rho_1|\rho_2 && \text{iff } \pi_{i,j} \models_K \rho_1 \vee \pi_{i,j} \models_K \rho_2\\
&\pi_{i,j} \models_K \rho^* && \text{iff } i = j \vee \exists k > 0.\pi_{i,j} \models_K \rho^k
\end{aligned}$$

Several derived operators can be defined in order to facilitate the specification of properties. The trajectory operator $\mathsf{EG}_\rho\varphi$ and the invariance operator $\mathsf{AG}_\rho\varphi$ are the duals of $\mathsf{AF}_\rho\varphi$ and $\mathsf{EF}_\rho\varphi$, respectively. They express that for some (resp. each) path going out of a state, all of its prefixes satisfying ρ lead to states satisfying φ. The potential saturation operator $\mathsf{EG}^{\dashv}_{\rho}$ and the inevitable saturation operator $\mathsf{AG}^{\dashv}_{\rho}$ are the negations of the corresponding looping operators. They express that some (resp. each) path going out of a state may begin with at most a finite number of repetitions of intervals satisfying ρ. Fairness properties can be expressed in Ctrl by means of the formula $\neg\mathsf{EF}^{\infty}_{\rho}$, which forbids the existence of unfair infinite execution sequences (see [25] for examples).

Expressiveness. Ctrl is a natural extension of Ctl [16] in which the until operator U is not primitive, but can be described using Ctrl's EF operator as follows: $\mathsf{E}[\varphi_1 \;\mathsf{U}\; \varphi_2] = \mathsf{EF}_{\varphi_1^*}\, \varphi_2$. Other extensions of Ctl, such as Rctl [19] and RegCtl [20], keep the U operator primitive as in the original logic. Ctrl subsumes RegCtl, whose U operator indexed by a regular formula can be expressed in Ctrl as follows: $\mathsf{E}[\varphi_1 \;\mathsf{U}^{\rho}\; \varphi_2] = \mathsf{EF}_{\rho \,\&\, \varphi_1^*}\varphi_2$, where & denotes the intersection of regular formulas (its occurrence in EF can be translated concisely in terms of the other regular operators [25]). The subsumption of RegCtl is strict because the U operator of RegCtl cannot describe an infinite concatenation of intervals satisfying a regular formula ρ, as specified by the $\mathsf{EF}^{\infty}_{\rho}$ operator of Ctrl. In [20] it is shown that RegCtl is more expressive than Rctl [19], the extension of Ctl with regular expressions underlying the Sugar [26] specification language; consequently, Rctl is also subsumed by Ctrl.

The potential looping operator EF^{∞} is able to capture the acceptance condition of Büchi automata, making Ctrl more expressive than Ltl [15]. Assuming that p characterizes the accepting states in a Büchi automaton (represented as a Ks), the formula $\mathsf{EF}^{\infty}_{\mathsf{true}^*.p.\mathsf{true}}$ expresses the existence of an infinite sequence passing infinitely often through an accepting state, where the $p.\mathsf{true}$ regular subformula avoids infinite sequences consisting of a single p-state. Although EF^{∞} does not allow a direct translation of the Ltl operators, it may serve as an intermediate form for model checking Ltl formulas; in this respect, this operator is similar to the "never claims" used for specifying properties in the early versions of the Spin model checker [27]. Since Ctl and Ltl are uncomparable w.r.t. their expressive power [16], it turns out that they are strictly subsumed by Ctrl. In fact, the Ctrl fragment containing the boolean connectors and the temporal operators EF and EF^{∞} is the state-based counterpart of Pdl-Δ [22], which has been shown to be more expressive than Ctl* [28].

3 On-the-Fly Model Checking

Our method for evaluating Ctrl formulas on Kss on-the-fly relies on a translation from Ctrl to HmlR and on the connection with an existing on-the-fly model checker for HmlR specifications on Ltss. In this section we briefly describe this translation by means of various examples of Ctrl temporal operators (see [25] for formal definitions and proofs). We also illustrate the functioning of the HmlR model checker, which rephrases the verification problem as the local resolution of a boolean equation system (Bes).

3.1 Translation from CTRL to HMLR

We consider as running example the following formula, stating that after every sequence matching $(p|q)^*.r$, either an r-state is eventually reached via a sequence satisfying $((p^*.q)|r^*)^*.q^*$, or p and q alternate along an infinite sequence:

$$\mathsf{AG}_{(p|q)^*.r}(\mathsf{AF}_{((p^*.q)|r^*)^*.q^*}r \lor \mathsf{EF}^{\infty}_{\mathsf{true}^*.p.\mathsf{true}^*.q})$$

This formula is neither expressible in Ctl (because of the EF^{∞} subformula), nor in Ltl (because of the nested $*$-operators). The translation from a Ctrl formula to a HmlR specification comprises three phases:

– The Ctrl formula is turned into a regular equation system (Res), which is a list of fixed point equation blocks interpreted on the Ks, having propositional variables in their left-hand sides and Ctrl state formulas in their right-hand sides. Ress are the state-based counterparts of PdlR (Pdl *with recursion*) specifications used as intermediate formalism for model checking regular alternation-free μ-calculus formulas [29] on Ltss.
– Each equation block in the Res is subsequently refined into a modal equation system (Mes) by eliminating all occurrences of regular operators contained in the regular formulas indexing the Ctrl operators. This is done by applying various transformations on the Res equations, according to the kind of temporal operators present in their right-hand sides.
– Finally, the resulting Mes is converted into a HmlR specification by replacing each occurrence of Ctrl temporal operator (now indexed by a state formula) with a Hml formula having the same interpretation on the Lts corresponding to the Ks.

The Ctrl formula above is translated into the following Res (μ and ν denote minimal and maximal fixed point equations, respectively):

$$\{X_1 \overset{\nu}{=} \mathsf{AG}_{(p|q)^*.r}X_2, X_2 \overset{\nu}{=} Y_1 \lor Z_1\}.\{Y_1 \overset{\mu}{=} \mathsf{AF}_{((p^*.q)|r^*)^*.q^*}r\}.\{Z_1 \overset{\nu}{=} \mathsf{EF}_{\mathsf{true}^*.p.\mathsf{true}^*.q}Z_1\}$$

We explain below how this Res is refined into a Mes by applying the transformations specific to each temporal operator, and we also show how the Ks and the Mes are converted into an Lts and a HmlR specification, respectively.

***Operators* EF_ρ *and* AG_ρ.** The Ctrl formula $\mathsf{AG}_\rho\varphi$ is the state-based counterpart of the Pdl modality $[\rho]\varphi$, and therefore Pdl-like identities hold about the distributivity of the AG_ρ operator over the regular operators contained in ρ:

$$\mathsf{AG}_{\rho_1.\rho_2}\varphi = \mathsf{AG}_{\rho_1}\mathsf{AG}_{\rho_2}\varphi \quad \mathsf{AG}_{\rho_1|\rho_2}\varphi = \mathsf{AG}_{\rho_1}\varphi \land \mathsf{AG}_{\rho_2}\varphi \quad \mathsf{AG}_{\rho_1^*}\varphi = \varphi \land \mathsf{AG}_{\rho_1}\mathsf{AG}_{\rho_1^*}\varphi$$

Dual identities are valid for $\mathsf{EF}_\rho\varphi$, which corresponds to the Pdl modality $\langle\rho\rangle\varphi$. A repeated application of these identities to the equations of the first block of the Res above allows to eliminate all occurrences of regular operators, leading to the following Mes block:

$$\{X_1 \overset{\nu}{=} X_3 \land X_4, X_2 \overset{\nu}{=} Y_1 \lor Z_1, X_3 \overset{\nu}{=} \mathsf{AG}_r X_2, X_4 \overset{\nu}{=} \mathsf{AG}_p X_1 \land \mathsf{AG}_q X_1\}$$

This transformation introduces a linear increase in size of the Mes w.r.t. the Res. Note that additional equations were inserted in order to avoid nested occurrences of temporal operators; this is necessary for keeping the size of the final Bes linear w.r.t. the size of the Mes and of the Ks.

***Operators* AF_ρ *and* EG_ρ.** The AF_ρ operator does not satisfy the identities of EF_ρ, and thus the regular operators occurring in ρ cannot be eliminated simply by applying substitutions. The procedure we propose for expanding AF_ρ operators consists of the three steps below (a dual procedure holds for expanding EG_ρ operators). Without loss of generality, we assume that the Res block contains a single equation with an AF_ρ operator in its right-hand side.

(a) The equation block containing AF_ρ is first converted to *potentiality form* by replacing AF with EF and eliminating all occurrences of regular operators using the identities associated to EF. This operation does not preserve the semantics of the initial block, but we will take care to restore it at step (c). For the second block of our example Res, this yields the following Mes:

$$\{\ Y_1 \overset{\mu}{=} Y_2 \vee Y_3,\ \ Y_2 \overset{\mu}{=} Y_4 \vee Y_5,\ \ Y_3 \overset{\mu}{=} Y_6 \vee Y_7,\ \ Y_4 \overset{\mu}{=} r,\ \ Y_5 \overset{\mu}{=} \mathsf{EF}_q Y_2,$$
$$Y_6 \overset{\mu}{=} Y_8 \vee Y_9,\ \ Y_7 \overset{\mu}{=} Y_1 \vee Y_{10},\ \ Y_8 \overset{\mu}{=} \mathsf{EF}_q Y_1,\ \ Y_9 \overset{\mu}{=} \mathsf{EF}_p Y_6,\ \ Y_{10} \overset{\mu}{=} \mathsf{EF}_r Y_7\ \}$$

(b) The resulting Mes is further transformed to *guarded* potentiality form (Gpf) by eliminating all occurrences of unguarded propositional variables (not preceded by an EF operator) in the right-hand sides of equations. This is done by considering each equation $Y_i \overset{\mu}{=} \varphi_i$, by replacing with φ_i all unguarded occurrences of Y_i in other equations, and eliminating the possible self-recursive unguarded occurrences found on the way using the absorption property $Y_j \overset{\mu}{=} Y_j \vee \varphi_j \equiv Y_j \overset{\mu}{=} \varphi_j$ [25]. When brought to Gpf and simplified (by deleting redundant variable occurrences using idempotency of disjunction, dropping identical equations, and renumbering variables), the Mes block becomes:

$$Y_1 \overset{\mu}{=} \mathsf{EF}_p Y_3 \vee \mathsf{EF}_q Y_1 \vee \mathsf{EF}_q Y_2 \vee \mathsf{EF}_r Y_1 \vee Y_4,\ \ Y_2 \overset{\mu}{=} \mathsf{EF}_q Y_2 \vee Y_4,\ \ Y_3 \overset{\mu}{=} \mathsf{EF}_p Y_3 \vee \mathsf{EF}_q Y_1,\ \ Y_4 \overset{\mu}{=} r$$

A Mes in Gpf is similar to the equation system defining the derivatives of regular expressions [30].

(c) The Mes in Gpf is finally *determinized* in order to retrieve the interpretation of the original Res block containing the AF operator. This is done by considering *meta-variables* (i.e., sets of propositional variables) holding at a state s and determining, for each combination of atomic propositions that may hold at s, the meta-variables that should be satisfied by the successors of s. We show below two equations obtained by determinizing the Mes above ($Y_{\{1,2\}}$ stands for the meta-variable $\{Y_1, Y_2\}$, and similarly for the others):

$$Y_{\{1\}} \overset{\mu}{=} \mathsf{AF}_p Y_{\{3\}} \vee \mathsf{AF}_q Y_{\{1,2\}} \vee \mathsf{AF}_r Y_{\{1\}} \vee \mathsf{AF}_{p\wedge q} Y_{\{1,2,3\}} \vee \mathsf{AF}_{p\wedge r} Y_{\{1,3\}} \vee \mathsf{AF}_{q\wedge r} Y_{\{1,2\}} \vee$$
$$\mathsf{AF}_{p\wedge q\wedge r} Y_{\{1,2,3\}} \vee Y_{\{4\}},\ \ Y_{\{3\}} \overset{\mu}{=} \mathsf{AF}_p Y_{\{3\}} \vee \mathsf{AF}_q Y_{\{1\}} \vee \mathsf{AF}_{p\wedge q} Y_{\{1,3\}}$$

The rhs of the equations defining the meta-variables $Y_{\{1,2\}}$, $Y_{\{1,3\}}$, and $Y_{\{1,2,3\}}$ is identical to the one defining $Y_{\{1\}}$. After further simplifications (induced by the implication $\mathsf{AF}_{p \wedge q}\varphi \Rightarrow \mathsf{AF}_q\varphi$) we obtain the final MES:

$$Y_{\{1\}} \overset{\mu}{=} \mathsf{AF}_p Y_{\{3\}} \vee \mathsf{AF}_q Y_{\{1\}} \vee \mathsf{AF}_r Y_{\{1\}} \vee Y_{\{4\}}, \quad Y_{\{3\}} \overset{\mu}{=} \mathsf{AF}_p Y_{\{3\}} \vee \mathsf{AF}_q Y_{\{1\}}, \quad Y_{\{4\}} \overset{\mu}{=} r$$

The determinization step is similar to the subset construction used for determinizing finite automata [31].

The final MES produced after expanding a RES block containing an AF_ρ operator has in the worst-case a size exponential w.r.t. the size of ρ; however the temporal formulas encountered in practice are far from reaching this bound. In particular, when ρ is "deterministic", i.e., for each equation of the corresponding MES in GPF, the atomic propositions indexing the EF operators in the right-hand side are disjoint (e.g., p and q disjoint in the MES above), the resulting determinized MES has a size linear w.r.t. ρ.

Operators EF^∞_ρ***,*** $\mathsf{AG}^\dashv_\rho$***,*** AF^∞_ρ***, and*** $\mathsf{EG}^\dashv_\rho$***.*** The infinite iteration operators (and their saturation duals) must be translated into RESs with alternation depth 2, because they involve two mutually recursive minimal and maximal fixed points. The third RES equation block of our running example would translate as follows:

$$\{Z_0 \overset{\nu}{=} Z_1\}.\{Z_1 \overset{\mu}{=} \mathsf{EF}_{\mathsf{true}^*.p.\mathsf{true}^*.q} Z_0\}$$

However, given the very simple structure of the first equation block, we can abusively merge the two blocks into a minimal fixed point one, expand the regular subformula using the EF substitutions, and mark the Z_0 variable such that the original semantics of the equation blocks can be restored during the resolution of the underlying BES (see Sec. 3.2). The AF^∞_ρ operator is expanded in a similar manner, and the saturation operators $\mathsf{AG}^\dashv_\rho$ and $\mathsf{EG}^\dashv_\rho$ are handled dually.

Moving from the state-based to the action-based setting. In order to apply a HMLR model checker as verification back-end for CTRL, we need to interpret formulas on LTSs instead of KSs. A KS can be converted to an LTS by migrating all the atomic propositions valid at each state of the KS on the actions labeling the transitions going out from that state in the LTS [16]. This conversion is succinct (it keeps the same state set and transition relation) and can be performed on-the-fly during an incremental construction of the KS. The MES produced from a CTRL formula can be turned into a HMLR specification by replacing basic CTRL formulas with HML modalities having the same interpretation on the LTS corresponding to the KS:

$$p = \langle p\rangle\mathsf{true} \qquad \mathsf{EF}_p X = \langle p\rangle X \qquad \mathsf{AG}_p X = [p]X$$
$$\mathsf{AF}_p X = \langle p\rangle\mathsf{true} \wedge [\mathsf{true}]X \qquad \mathsf{EG}_p X = \langle p\rangle\mathsf{true} \Rightarrow \langle\mathsf{true}\rangle X$$

These replacements increase by at most a linear factor the size of the HMLR specification w.r.t. the MES.

3.2 BES Encoding and Local Resolution

The on-the-fly model checking of the HmlR specification produced from a Ctrl formula on the Lts corresponding to a Ks can be rephrased as the local resolution of a Bes [23,32], which can be carried out using graph-based algorithms [33,34,24]. Figure 1 illustrates the evaluation of a Ctrl infinite looping operator on a Ks. For simplicity, we show the verification by considering directly the Mes (produced as indicated in Sec. 3.1) and the Ks instead of the corresponding HmlR specification and Lts.

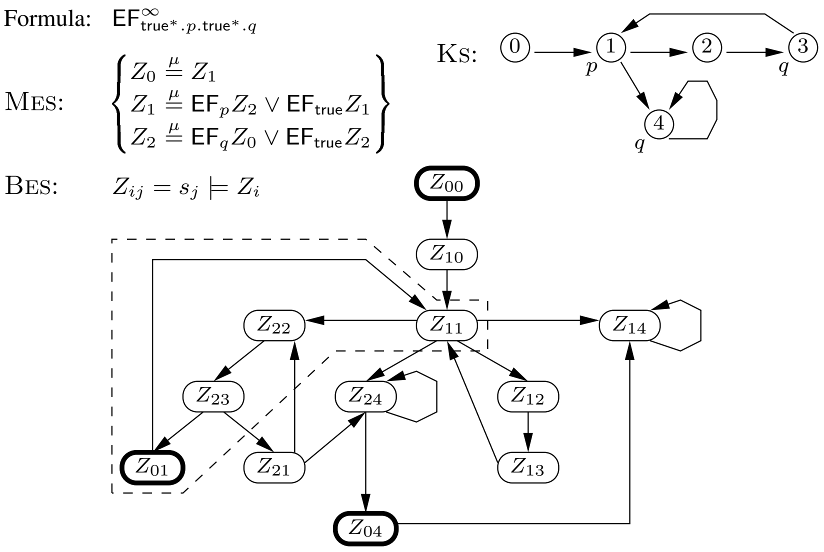

Formula: $\mathsf{EF}^{\infty}_{\mathsf{true}*.p.\mathsf{true}*.q}$
Mes: $\left\{ \begin{array}{l} Z_0 \stackrel{\mu}{=} Z_1 \\ Z_1 \stackrel{\mu}{=} \mathsf{EF}_p Z_2 \vee \mathsf{EF}_{\mathsf{true}} Z_1 \\ Z_2 \stackrel{\mu}{=} \mathsf{EF}_q Z_0 \vee \mathsf{EF}_{\mathsf{true}} Z_2 \end{array} \right\}$
Bes: $Z_{ij} = s_j \models Z_i$
Ks:
0
1
2
3
4
p
q
q
Z_{00}
Z_{10}
Z_{11}
Z_{22}
Z_{14}
Z_{23}
Z_{24}
Z_{12}
Z_{01}
Z_{21}
Z_{13}
Z_{04}

Fig. 1. Evaluation of a EF^{∞} formula. The underlying Bes is obtained by making a product between the Mes and the Ks. It is represented here by its boolean graph, which is explored on-the-fly by the A4_{cyc} algorithm.

The Bes encoding the model checking problem is disjunctive, and could be solved using the memory-efficient algorithm A4 proposed in [24]. If the EF^{∞} formula is false, the solution of the Mes is also false, since by abusively switching the sign from ν to μ we obtained an equation block with a "smaller" interpretation. If the formula is true, the Ks contains a cycle going through a state satisfying the marked variable Z_0; this kind of cycle is detected in linear-time by the A4_{cyc} algorithm [35], which records that all states on the cycle satisfy EF^{∞}, thus restoring the original meaning of the formula.

Complexity. The complexity of our Ctrl model checking procedure is summarized in the table below. The EF_ρ and $\mathsf{EF}^{\infty}_\rho$ operators, together with their respective duals AG_ρ and $\mathsf{AG}^{\dashv}_\rho$, are evaluated in linear-time w.r.t. the size of ρ and the size of the Ks. Moreover, the evaluation of these operators stores only the states (and not the transitions) of the Ks, thanks to the memory-efficient algorithms A4 [24] and A4$_{cyc}$ [35] dedicated to disjunctive and conjunctive Bess. This fragment of Ctrl is the state-based counterpart of Pdl-Δ [22]. The linear-time evaluation of the EF_ρ^∞ operator allows an efficient detection of complex cycles, such as those characterizing oscillation properties [9]. EF_ρ^∞ is also useful for capturing fairness properties in concurrent systems, such as the existence of complex unfair executions in resource locking protocols [36].

The AF_ρ operator and its dual EG_ρ are evaluated in linear-time only when the regular subformula ρ is deterministic. In general, these operators are evaluated in exponential-time w.r.t. the size of ρ (because of the determinization step) but still in linear-time w.r.t. the Ks size. In practice, the size of temporal formulas is much smaller than the size of Kss, which reduces the impact of the factor $2^{|\rho|}$ on the total cost of model checking. Finally, the AF_ρ^∞ operator and its dual $\mathsf{EG}_\rho^\dashv$ are evaluated in linear-time when ρ is deterministic (using a symmetric version of the A4$_{cyc}$ algorithm); in the general case, these operators are evaluated in doubly exponential-time w.r.t. the size of ρ and in quadratic-time w.r.t. the Ks size, by applying local resolution algorithms for Bess with alternation depth 2 [34]. This complexity seems difficult to lower, since the Bess produced by translating these operators have a general shape (arbitrary nesting of disjunctions and conjunctions in the right-hand sides of equations).

Ctrl operator	Model checking complexity	
	ρ deterministic	ρ nondeterministic
EF_ρ AG_ρ	$O(\lvert\rho\rvert \cdot (\lvert S\rvert + \lvert T\rvert))$	
AF_ρ EG_ρ	$O(\lvert\rho\rvert \cdot (\lvert S\rvert + \lvert T\rvert))$	$O(2^{\lvert\rho\rvert} \cdot (\lvert S\rvert + \lvert T\rvert))$
EF_ρ^∞ $\mathsf{AG}_\rho^\dashv$	$O(\lvert\rho\rvert \cdot (\lvert S\rvert + \lvert T\rvert))$	
AF_ρ^∞ $\mathsf{EG}_\rho^\dashv$	$O(\lvert\rho\rvert \cdot (\lvert S\rvert + \lvert T\rvert))$	$O(2^{2^{\lvert\rho\rvert}} \cdot (\lvert S\rvert + \lvert T\rvert)^2)$

4 Implementation and Use

We implemented the model checking procedure for Ctrl described in Section 3 by reusing as much as possible the on-the-fly verification technology available in the Cadp toolbox [13] for concurrent asynchronous systems. This section presents the architecture of our Ctrl model checker and illustrates its use for analyzing genetic regulatory networks.

4.1 An On-the-Fly Model Checker for CTRL

The tools of Cadp[1] (*Construction and Analysis of Distributed Processes*) [13] operate on labeled transition systems (Ltss), which are represented either explicitly (by their list of transitions) as compact binary files encoded in the Bcg (*Binary Coded Graphs*) format, or implicitly (by their successor function) as C programs compliant with the Open/Cæsar interface [37]. Cadp contains the on-the-fly model checker Evaluator [29], which evaluates regular alternation-free μ-calculus ($L\mu_1^{reg}$) formulas on implicit Ltss. The tool works by translating the verification problem in terms of the local resolution of a Bes, which is

[1] `http://www.inrialpes.fr/vasy/cadp`

done using the algorithms available in the generic CÆSAR_SOLVE library [24]. EVALUATOR 3.6 uses HMLR as intermediate language: $L\mu_1^{reg}$ formulas are translated into HMLR specifications, whose evaluation on implicit LTSs is encoded as a local BES resolution. It generates examples and counterexamples illustrating the truth value of formulas, and is also equipped with macro-definition mechanisms allowing the creation of reusable libraries of derived temporal operators.

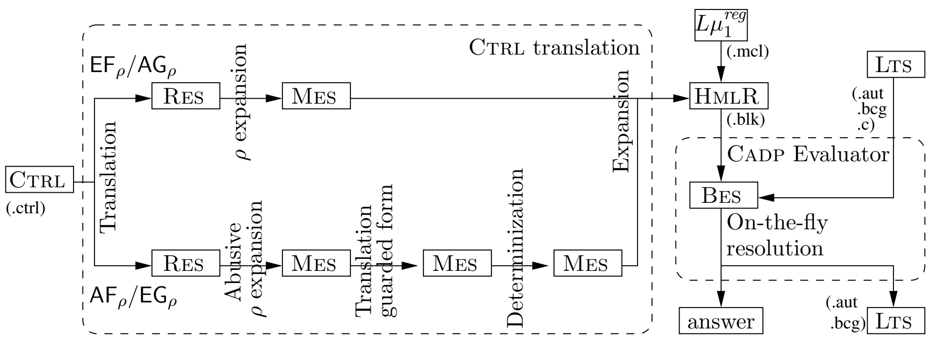


Fig. 2. CTRL translator and its connection to the EVALUATOR model checker

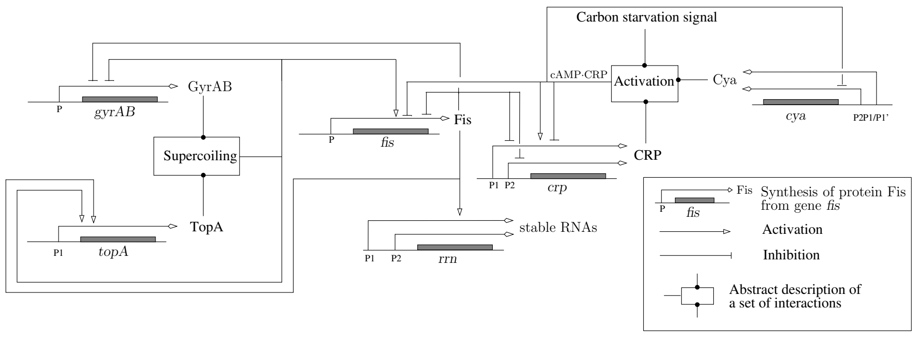
In order to reuse the model checking features of EVALUATOR 3.6, we had the choice of translating CTRL formulas either to $L\mu_1^{reg}$ formulas, or to HMLR specifications. We adopted the second solution because it leads to a more succinct translation and avoids the translation step from $L\mu_1^{reg}$ to HMLR present in EVALUATOR. This technical choice motivated the definition of the translation from CTRL to HMLR in the first place. The architecture of the CTRL translator (about 12,000 lines of code) is shown in Figure 2. The tool takes as input a CTRL state formula and translates it to a MES following the phases described in Section 3.1, which are different for the EF_ρ and AF_ρ operators. The MES obtained is then converted into a HMLR specification by replacing basic CTRL operators with HML modalities. The resulting HMLR specification is directly given as input to EVALUATOR 3.6, together with the LTS corresponding to the KS. The translator from CTRL to HMLR has been completely implemented using the compiler construction technology proposed in [38].

4.2 Verification of Genetic Regulatory Networks

CTRL has been used for the analysis of so-called *genetic regulatory networks* (GRNs), which consist of genes, proteins, small molecules and their mutual interactions that together control different functions of the cell. In order to better understand how a specific dynamic behavior emerges from these interactions, and the role of each component in the network, a wide variety of mathematical formalisms are available [3].

Due to limited availability of numerical values for the kinetic parameters and the molecular concentrations, some mathematical formalisms are difficult to apply in practice. This has motivated the use of a special class of *piecewise-linear* (Pl) *differential equation* models, originally introduced by [39]. Using Pl models, the qualitative dynamics of the high-dimensional systems are relatively straightforward to analyze, using inequality constraints on the parameters rather than exact numerical values [40,41]. Discrete abstractions can be used to convert the continuous dynamics of the Pl systems into state transition graphs (Stgs) [40] that are formally equivalent to Kss. The atomic propositions describe, among other things, the concentration bounds defining a region and the trend of the variables inside regions. The generation of the Stg from the Pl model has been implemented in the computer tool Gna (*Genetic Network Analyzer*) [6], which is able to export the graph to standard model checkers like NuSmv [12] and Cadp [13] in order to use formal verification.

We analyse here the carbon starvation response network of *E. coli* (illustrated below), using a Pl model proposed in [42], with focus on the nutrient upshift after a period of starvation, leading to exponential growth of the bacterial population. The dynamics of the system are described by 6 coupled Pl differential equations, and 48 inequality constraints on the parameter values.



The generated graph has 743 states and contains one terminal cycle corresponding to a (damped) oscillation of some of the protein concentrations and the concentration of stable Rnas (which are transcribed from the *rrn* operons). We expressed this property using the four Ctrl formulas below, where *inTermCycle* is an atomic proposition indicating that a state is part of the terminal cycle. Similarly, *dec_rrn* (*inc_rrn*) represent a decreasing (increasing) concentration of stable Rnas (ρ^+ stands for $\rho.\rho^*$).

N.	Ctrl formula	Answer	Time
1.	$\mathsf{EF}_{\mathsf{true}^*}\,\mathsf{AF}_{inTermCycle^+.(inc\_rrn^+.dec\_rrn^+)^+}\,\mathsf{true}$	false	3 sec
2.	$\mathsf{EF}_{\mathsf{true}^*}\,\mathsf{EF}^{\infty}_{inTermCycle^+.(inc\_rrn^+.dec\_rrn^+)^+}$	true	1 sec
3.	$\mathsf{AG}_{\mathsf{true}^*}\,\mathsf{EF}^{\infty}_{inTermCycle^+.(inc\_rrn^+.dec\_rrn^+)^+}$	false	1 sec
4.	$\mathsf{AG}_{true^*.inc\_Fis^+.dec\_Crp^+.inTermCycle}\,\mathsf{EF}^{\infty}_{inTermCycle^+.(inc\_rrn^+.dec\_rrn^+)^+}$	true	2 sec

Formula 1 fails, indicating that an oscillation of stable Rnas is not inevitable once the system has reached the terminal cycle. Formula 2, obtained by replacing the AF operator with an EF^{∞}, is valid on the graph, showing the existence of an infinite oscillation of the stable RNAs. Formula 3 is stricter, stating that all paths in the graph lead to the terminal cycle with an oscillation of stable RNAs. Formula 4 forces the model checker to consider the oscillation only on the paths satisfying the restriction that an increase of the Fis concentration is followed by a decrease of the Crp concentration before arriving at the terminal cycle.

The use of regular expressions in the CTRL formulas above clearly outlines the convenience of being able to characterize a sequence of events. Due to the presence of nested iteration operators, these properties cannot be expressed using standard temporal logics such as CTL or LTL. In addition, the $\mathsf{EF}_{\rho}^{\infty}$ operator enables a natural formulation of infinite repetitions of sequences defined by ρ, such as those corresponding to the oscillation in the *E. coli* example.

5 Conclusion and Future Work

Applications of model checking in system biology have demonstrated its usefulness for understanding the dynamic behaviour of regulatory networks in living cells, but also pointed out certain limitations in expressiveness and user-friendliness. Our work aims at alleviating these limitations in order to promote the practical usage of model checking in the bioinformatics and systems biology communities. CTRL extends CTL with regular expressions and fairness operators, allowing a natural and concise description of typical properties of biological interest, such as the presence of multistability or oscillations. We were able to reduce the development effort and to obtain an on-the-fly model checker for CTRL by defining and implementing a translation from CTRL to HMLR, and by reusing the verification and diagnostic generation features of the EVALUATOR 3.6 model checker of the CADP toolbox.

In this paper, we have employed CTRL for the verification of dynamic properties of GRNs modeled by (but not limited to) piecewise-linear differential equations. The continuous dynamics of these models, by defining appropriate discrete abstractions, can be converted into discrete state transition graphs that are formally equivalent to KSs. The computer tool GNA is able to generate the state transition graphs and export them as LTSs to CADP. CTRL can be combined with many of the other approaches proposed for the application of formal verification tools to biological regulatory networks [4,5,6,7,8,9,10].

We plan to continue our work on several directions. First, we will extend the CÆSAR_SOLVE [24] library of CADP with resolution algorithms handling BESs of alternation depth 2 [34] in order to obtain an on-the-fly evaluation of the $\mathsf{AF}_{\rho}^{\infty}$ operator when the regular formula ρ is nondeterministic. Second, the translation from CTRL to HMLR can be optimized by adding static analysis features on the GNA atomic propositions in order to reduce the size of the HMLR specifications produced. Third, a distributed version of the CTRL model checker can be obtained by coupling it with the distributed BES resolution algorithms proposed in [43,44]. Fourth, we will develop pattern-based tools to help non-expert users specify queries for the analysis of biological networks [45].

Acknowledgements. This research was funded by the Ec-Moan project no. 043235 of the Fp6-Nest-Path-Com European program. Pedro T. Monteiro is also supported by the Fct program (PhD grant Sfrh/Bd/32965/2006).

References

1. Fisher, J., Henzinger, T.A.: Executable cell biology. Nature Biotechnology 25(11), 1239–1250 (2007)
2. Regev, A., Shapiro, E.: Cells as computation. Nature 419(6905), 343 (2002)
3. de Jong, H.: Modeling and simulation of genetic regulatory systems: A literature review. J. of Computational Biology 9(1), 67–103 (2002)
4. Antoniotti, M., Policriti, A., Ugel, N., Mishra, B.: Model building and model checking for biochemical processes. Cell Biochemistry and Biophysics 38(3), 271–286 (2003)
5. Barnat, J., Brim, L., Cerná, I., Drazan, S., Safranek, D.: Parallel model checking large-scale genetic regulatory networks with DiVinE. In: FBTC 2007. ENTCS, vol. 194 (2008)
6. Batt, G., Ropers, D., de Jong, H., Geiselmann, J., Mateescu, R., Page, M., Schneider, D.: Validation of qualitative models of genetic regulatory networks by model checking: Analysis of the nutritional stress response in *Escherichia coli*. Bioinformatics 21 (Suppl. 1), i19–i28 (2005)
7. Bernot, G., Comet, J.-P., Richard, A., Guespin, J.: Application of formal methods to biological regulatory networks: Extending Thomas' asynchronous logical approach with temporal logic. J. of Theoretical Biology 229(3), 339–348 (2004)
8. Calder, M., Vyshemirsky, V., Gilbert, D., Orton, R.: Analysis of signalling pathways using the PRISM model checker. In: CMSB 2005, pp. 79–90 (2005)
9. Chabrier-Rivier, N., Chiaverini, M., Danos, V., Fages, F., Schächter, V.: Modeling and querying biomolecular interaction networks. TCS 325(1), 25–44 (2004)
10. Fisher, J., Piterman, N., Hajnal, A., Henzinger, T.A.: Predictive modeling of signaling crosstalk during *C. elegans* vulval development. PLoS Computational Biology 3(5), e92 (2007)
11. Batt, G., Bergamini, D., de Jong, H., Gavarel, H., Mateescu, R.: Model checking genetic regulatory networks using GNA and CADP. In: Graf, S., Mounier, L. (eds.) SPIN 2004. LNCS, vol. 2989, pp. 158–163. Springer, Heidelberg (2004)
12. Cimatti, A., Clarke, E., Giunchiglia, F., Roveri, M.: NuSMV: a new symbolic model checker. STTT 2(4), 410–425 (2000)
13. Garavel, H., Lang, F., Mateescu, R., Serwe, W.: CADP 2006: A toolbox for the construction and analysis of distributed processes. In: Damm, W., Hermanns, H. (eds.) CAV 2007. LNCS, vol. 4590, pp. 158–163. Springer, Heidelberg (2007)
14. 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)
15. Manna, Z., Pnueli, A.: The Temporal Logic of Reactive and Concurrent Systems. Specification, vol. I. Springer, Heidelberg (1992)
16. Clarke, E.M., Grumberg, O., Peled, D.A.: Model Checking. MIT Press, Cambridge (2000)
17. Emerson, E.A., Halpern, J.Y.: Sometimes and not never revisited: On branching versus linear time. In: POPL 1983, pp. 127–140 (January 1983)

18. Kozen, D.: Results on the propositional μ-calculus. TCS 27, 333–354 (1983)
19. Beer, I., Ben-David, S., Landver, A.: On-the-fly model checking of RCTL formulas. In: Y. Vardi, M. (ed.) CAV 1998. LNCS, vol. 1427, pp. 184–194. Springer, Heidelberg (1998)
20. Brázdil, T., Cerná, I.: Model checking of RegCTL. Computers and Artificial Intelligence 25(1) (2006)
21. Fischer, M.J., Ladner, R.E.: Propositional dynamic logic of regular programs. JCSS 18(2), 194–211 (1979)
22. Streett, R.: Propositional dynamic logic of looping and converse. Information and Control (1982)
23. Larsen, K.G.: Proof systems for Hennessy-Milner logic with recursion. In: Dauchet, M., Nivat, M. (eds.) CAAP 1988. LNCS, vol. 299, pp. 215–230. Springer, Heidelberg (1988)
24. Mateescu, R.: CÆSAR_SOLVE: A generic library for on-the-fly resolution of alternation-free boolean equation systems. STTT 8(1), 37–56 (2006)
25. Mateescu, R., Monteiro, P.T., Dumas, E., Mateescu, R.: Computation tree regular logic for genetic regulatory networks. Research Report RR-6521, INRIA (2008)
26. Beer, I., Ben-David, S., Eisner, C., Fisman, D., Gringauze, A., Rodeh, Y.: The temporal logic Sugar. In: Berry, G., Comon, H., Finkel, A. (eds.) CAV 2001. LNCS, vol. 2102, pp. 363–367. Springer, Heidelberg (2001)
27. Holzmann, G.: The SPIN Model Checker – Primer and Reference Manual. Addison-Wesley, Reading (2003)
28. Wolper, P.: A translation from full branching time temporal logic to one letter propositional dynamic logic with looping (published manuscript, 1982)
29. Mateescu, R., Sighireanu, M.: Efficient on-the-fly model-checking for regular alternation-free mu-calculus. SCP 46(3), 255–281 (2003)
30. Brzozowski, J.A.: Derivatives of regular expressions. JACM 11(4), 481–494 (1964)
31. Aho, A.V., Sethi, R., Ullman, J.D.: Compilers: Principles, Techniques and Tools. Addison-Wesley, Reading (1986)
32. Cleaveland, R., Steffen, B.: A linear-time model-checking algorithm for the alternation-free modal mu-calculus. FMSD 2(2), 121–147 (1993)
33. Andersen, H.R.: Model checking and boolean graphs. TCS 126(1), 3–30 (1994)
34. Vergauwen, B., Lewi, J.: Efficient local correctness checking for single and alternating boolean equation systems. In: Shamir, E., Abiteboul, S. (eds.) ICALP 1994. LNCS, vol. 820, pp. 304–315. Springer, Heidelberg (1994)
35. Mateescu, R., Thivolle, D.: A model checking language for concurrent value-passing systems. In: Cuellar, J., Maibaum, T.S.E. (eds.) FM 2008. LNCS, vol. 5014, pp. 148–164. Springer, Heidelberg (2008)
36. Arts, T., Earle, C.B., Derrick, J.: Development of a verified Erlang program for resource locking. STTT 5(2–3), 205–220 (2004)
37. Garavel, H.: OPEN/CÆSAR: An open software architecture for verification, simulation, and testing. In: Steffen, B. (ed.) TACAS 1998. LNCS, vol. 1384, pp. 68–84. Springer, Heidelberg (1998)
38. Garavel, H., Lang, F., Mateescu, R.: Compiler construction using LOTOS NT. In: Horspool, R.N. (ed.) CC 2002. LNCS, vol. 2304, pp. 9–13. Springer, Heidelberg (2002)
39. Glass, L., Kauffman, S.A.: The logical analysis of continuous non-linear biochemical control networks. J. of Theoretical Biology 39(1), 103–129 (1973)
40. Batt, G., de Jong, H., Page, M., Geiselmann, J.: Symbolic reachability analysis of genetic regulatory networks using discrete abstractions. Automatica 44(4), 982–989 (2008)

41. 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)
42. Ropers, D., de Jong, H., Page, M., Schneider, D., Geiselmann, J.: Qualitative simulation of the carbon starvation response in *Escherichia coli.* Biosystems 84(2), 124–152 (2006)
43. Joubert, C., Mateescu, R.: Distributed local resolution of boolean equation systems. In: PDP 2005. IEEE Computer Society, Los Alamitos (2005)
44. Joubert, C., Mateescu, R.: Distributed on-the-fly model checking and test case generation. In: Valmari, A. (ed.) SPIN 2006. LNCS, vol. 3925, pp. 126–145. Springer, Heidelberg (2006)
45. Monteiro, P.T., Ropers, D., Mateescu, R., Freitas, A.T., de Jong, H.: Temporal logic patterns for querying dynamic models of cellular interaction networks. Bioinformatics (in press, 2008)