CTRL: Extension of CTL with Regular Expressions and Fairness Operators to Verify Genetic Regulatory Networks

Radu Mateescu[a], Pedro T. Monteiro[a,b], Estelle Dumas[a], Hidde de Jong[a]

[a] *INRIA Grenoble – Rhône-Alpes, Inovallée, Montbonnot, 38334 St Ismier Cedex, France*
[b] *IST/INESC-ID, 9 Rua Alves Redol, 1000-029 Lisbon, Portugal*

Abstract

Model checking has proven to be a useful analysis technique not only for concurrent systems, but also for genetic regulatory networks (GRNs). Applications of model checking in systems biology have revealed that temporal logics should be able to capture both branching-time and fairness properties (needed for specifying multistability and oscillation properties, respectively). At the same time, they should have a user-friendly syntax easy to employ by non-experts. In this paper, we define Computation Tree Regular Logic (CTRL), 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. We also develop a translation of CTRL into Hennessy-Milner Logic with Recursion (HMLR), 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*.

Key words:
genetic regulatory networks, model checking, systems biology, temporal logic, verification

NOTICE: this is the author's version of a work that was accepted for publication by Elsevier. Changes resulting from the publishing process, such as peer review, editing, corrections, structural formatting, and other quality control mechanisms may not be reflected in this document. Changes may have been made to this work since it was submitted for publication. A definitive version was subsequently published in Theoretical Computer Science, Volume 412, Issue 26, 10 June 2011, Pages 2854-2883. doi:10.1016/j.tcs.2010.05.009

1. Introduction

Formal 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 [40, 72] for reviews). Genetic regulatory networks (GRNs) consist of genes, proteins, small molecules, and their mutual interactions, which are involved in the control of cellular functions. Most of these networks are large and complex, thus defying our capacity to understand how the dynamic behaviour of the cell emerges from the structure of interactions. A large number of mathematical formalisms have been proposed to describe GRNs (see [32] for a survey), giving rise to models that can be mapped to Kripke structures. In the case of discrete models the mapping is usually direct, but for continuous models appropriately chosen discrete abstractions are required [4].

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. In particular, such properties can be formulated as queries in temporal logic, and verified by means of model checking algorithms on the Kripke structures. Examples of the kind of properties that biologists are interested in include the following:

- Is the basal glycerol production level combined with rapid closure of Fps1 sufficient to explain an initial glycerol accumulation after osmotic shock? [59]
- Once a cell has executed START, does it slip back into G1 phase and repeat START? Or rather, must it execute a FINISH to return to G1? [25]
- Does Shc phosphorylation exhibit a relative acceleration with decreasing EGF concentration and show a decline over time? [74]

Several applications of model checking exist in the bioinformatics and systems biology literature, *e.g.*, [5, 7, 8, 12, 13, 16, 18, 21, 22, 41, 42, 76].

In our previous work [10, 12], we have developed Genetic Network Analyzer (GNA), a tool for the qualitative simulation of GRNs, and connected it to state-of-the-art model checkers like NUSMV [27] and CADP [44].

Most of the above approaches express the properties of interest in classical temporal logics like CTL [28] and LTL [65]. The application to actual

biological systems brought a few properties of the network dynamics to the fore that are not easily expressed in these logics. For instance, questions about multistability are important in the analysis of biological regulatory networks [35, 79], but difficult (or impossible) to express in LTL. CTL is capable of dealing with branching time, important for multistability and other properties of non-deterministic models. However, it does not do a good job when faced with questions about cycles in a Kripke structure. Such cycles may correspond to sustained or damped oscillations in the concentration of molecular species, underlying cellular rhythms [25, 64]. CTL is not expressive enough to specify the occurrence of oscillations of indefinite length, a special kind of fairness property [12]. An obvious solution would be to consider CTL* [36] or the propositional μ-calculus [60], both of which subsume CTL and LTL; however, these powerful branching-time logics are complex to understand and use by non-experts. More generally, temporal logics have difficulties in expressing experimental observations, which often take the form of patterns of events corresponding to variations of system parameters (protein concentrations, their derivatives, etc.). 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. In order to achieve 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 [15] and RegCTL [19], whilst having a simpler syntax due to a different choice of primitive temporal operators, inspired from dynamic logics like PDL [39]. CTRL also subsumes CTL, LTL, and PDL-Δ [77], allowing in particular the concise expression of bistability and oscillation properties by using potentiality and fairness operators, respectively. The characterization of oscillations motivated the introduction of fairness operators in CTRL at the same level as the other temporal operators of the logic. These operators, derived from the infinite looping operator of PDL-Δ, provide a more direct description of the complex cycles underlying oscillations than the operators of other logics equipped with fairness, such as fair CTL [38]. Although CTRL was primarily designed for describing properties of regulatory networks in systems 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 attempt to avoid the effort of building a model checker from scratch by reusing as much as possible existing verification technology. We adopt as verification engine CADP [44], 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 [28], simply by migrating information from states to transition labels without changing the structure of the model, *i.e.*, keeping the same states and transition relations. The translation from CTRL to an action-based logic is carried out by considering as target language HMLR [63], an alternative equational representation of the modal μ-calculus. Since HMLR is accepted as input by the EVALUATOR 3.6 [69] model checker of CADP, the development of a translator from CTRL to HMLR results in the immediate availability of an on-the-fly model checker equipped with full diagnostic features (generation of examples and counterexamples).

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 point by analyzing a model of the GRN involved in the carbon starvation response of *E. coli*. The network consists of key global regulators of transcription that control each other's expression as well as the expression of a large number of other stress response genes. We use a qualitative model of this network in the form of nine coupled piecewise-linear differential equations developed in GNA [71, 73]. This model has been shown to have interesting stability and oscillation properties that are difficult to analyze by hand though, as the Kripke structures corresponding to the model simulations consist of 10^4-10^{10} states. We formulate increasingly

precise CTRL queries which clearly bring out the utility of the regular expressions and fairness operators of the language. The queries are verified by exporting the Kripke structure from GNA to CADP and invoking the CTRL model checker.

Paper outline. Section 2 defines the syntax and semantics of CTRL and discusses its expressiveness w.r.t. existing widely-used logics. Section 3.1 defines the regular equation systems (RESs), an intermediate equational form into which CTRL formulas will be translated. Sections 3.2 and 3.3 present the translations from CTRL to RESs and then to modal equation systems (MESs). Section 4 describes the on-the-fly model checking procedure for CTRL, indicates its complexity, and shows its implementation in connection with GNA and CADP. Section 5 illustrates its application on the example of *E. coli*. Section 6 provides some concluding remarks and directions for future work. Proofs of the translation from CTRL to MESs can be found in the supplementary material available online.

2. Syntax and semantics

2.1. Computation Tree Regular Logic

CTRL is interpreted on Kripke structures, which provide a natural formal description of concurrent systems, including biological regulatory networks. A Kripke structure 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 Kripke structure $K = \langle S, P, L, T, s_0 \rangle$, on which all formulas will be interpreted.

The syntax and semantics of CTRL are defined in Figure 1. The logic contains two kinds of entities: *state formulas* (noted φ) and *regular formulas*

(noted 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.

SYNTAX

State formulas:

$$
\begin{array}{rll}
\varphi ::= & p & \text{(atomic proposition)} \\
| & \neg\varphi \mid \varphi_1 \vee \varphi_2 & \text{(boolean connectors)} \\
| & \mathsf{EF}_\rho \varphi & \text{(potentiality)} \\
| & \mathsf{AF}_\rho \varphi & \text{(inevitability)} \\
| & \mathsf{EF}^\infty_\rho & \text{(potential looping)} \\
| & \mathsf{AF}^\infty_\rho & \text{(inevitable looping)}
\end{array}
$$

Regular formulas:

$$
\begin{array}{rll}
\rho ::= & \varphi & \text{(one-step interval)} \\
| & \rho_1.\rho_2 & \text{(concatenation)} \\
| & \rho_1|\rho_2 & \text{(choice)} \\
| & \rho^* & \text{(iteration 0 or more times)}
\end{array}
$$

SEMANTICS

State formulas:

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

Regular formulas:

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

Figure 1: Syntax and semantics of CTRL

The interpretation $[\![\varphi]\!]_K$ of a state formula denotes the set of states of the Kripke structure 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 a Kripke structure 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 $\mathsf{EF}_\rho\varphi$ (resp. inevitability formula $\mathsf{AF}_\rho\varphi$) iff some (resp. 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 (resp. the inevitable looping formula AF^∞_ρ) iff some (resp. 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. A Kripke structure K satisfies a state formula φ (notation $K \models \varphi$) iff $s_0 \in [\![\varphi]\!]_K$.

Figure 2 shows several derived operators on states and intervals 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 defined as duals of inevitability and potentiality operators, respectively, similarly to the their CTL counterparts (obtained by dropping the ρ formulas). 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 $\mathsf{AG}^\dashv_\rho$ express that some (resp. each) path going out of a state contains a prefix satisfying ρ^* such that no other larger prefix satisfies ρ^*; in other words, only a finite number of intervals satisfying ρ can be concatenated at the beginning of the path. The empty interval operator nil is defined as the iteration (0 or more times) of the false proposition; an interval satisfies the formula nil iff it contains a single state. The iteration (1 or more times) operator '+' is defined in the standard way; an interval satisfies ρ^+ iff it is the concatenation of (1 or more) intervals satisfying ρ.

To facilitate the manipulation of CTRL state formulas, we transform them in *positive normal form* (PNF) by propagating the negations downwards, using the rules in Figure 2, until they reach the atomic propositions p. For convenience, we also include in the set P the negations of all propositions p, as well as the boolean constants true and false. State formulas in PNF are

$\mathsf{true} = p \vee \neg p$	(true, $p \in P$)
$\mathsf{false} = \neg\mathsf{true}$	(false)
$\varphi_1 \wedge \varphi_2 = \neg(\neg\varphi_1 \vee \neg\varphi_2)$	(conjunction)
$\varphi_1 \Rightarrow \varphi_2 = \neg\varphi_1 \vee \varphi_2$	(implication)
$\varphi_1 \Leftrightarrow \varphi_2 = (\varphi_1 \Rightarrow \varphi_2) \wedge (\varphi_2 \Rightarrow \varphi_1)$	(equivalence)
$\mathsf{EG}_\rho\varphi = \neg\mathsf{AF}_\rho\neg\varphi$	(trajectory)
$\mathsf{AG}_\rho\varphi = \neg\mathsf{EF}_\rho\neg\varphi$	(invariance)
$\mathsf{EG}^\neg_\rho = \neg\mathsf{AF}^\infty_\rho$	(potential saturation)
$\mathsf{AG}^\neg_\rho = \neg\mathsf{EF}^\infty_\rho$	(inevitable saturation)
$\mathsf{nil} = \mathsf{false}^*$	(empty interval)
$\rho^+ = \rho.\rho^*$	(iteration 1 or more times)

Figure 2: Derived (boolean, temporal, and regular) operators of CTRL

thus composed of atomic propositions, disjunctions and conjunctions, and all primitive and derived CTRL temporal operators defined in Figures 1 and 2.

2.2. Examples of temporal properties

We illustrate below the use of CTRL operators for specifying typical temporal properties of biological regulatory networks. The analogy with properties of communication protocols and concurrent systems is made explicit through the terminology of safety, liveness and fairness.

Safety properties. Informally, these properties specify that "something bad never happens" during the functioning of the system. They can be expressed in CTRL by identifying the sequences of states corresponding to violations of the safe progression of execution, characterizing them using a regular formula ρ, and forbidding their existence in the Kripke structure by checking the formula $\mathsf{AG}_\rho\mathsf{false}$. For example, the CTRL formula below states that it is impossible to express cell cycle genes g_1 and g_2 in response to an external stress signal:

$$\mathsf{AG}_{\mathsf{true}^*.sig^+.(g_1|g_2)}\mathsf{false}$$

where the atomic proposition sig indicates the presence of the external signal, and g_1 and g_2 the expression of the cell cycle genes. This property can also be specified in CTL using two nested temporal operators:

$$\mathsf{AG}(sig \Rightarrow \neg\mathsf{E}[sig\ \mathsf{U}\ (g_1 \vee g_2)])$$

where $\mathsf{AG}\ \varphi = \neg\mathsf{E}[\mathsf{true}\ \mathsf{U}\ \neg\varphi]$ is the invariance operator of CTL.

Liveness properties. Informally, these properties specify that "something good eventually happens" during the functioning of the system. They can be expressed in CTRL by capturing the desirable sequences of states, characterizing them using a regular formula ρ, and expressing their potential or inevitable presence in the Kripke structure using the EF_ρ and AF_ρ operators, respectively. For instance, the CTRL formula below states that every time a particular nutrient *nut* is present in the medium, it will eventually be taken up and consumed by the cell, as witnessed by the expression of gene g_{nut} coding for an appropriate transporter. This may be preceded by the expression of one or more genes in the set G, responsible for the uptake and consumption of other nutrients:

$$\mathsf{AG}_{\mathsf{true}^*.nut}\mathsf{AF}_{(\mathsf{true}^*.G)^*.g_{nut}}\mathsf{true}$$

where the atomic proposition G indicates that one or more genes in the corresponding set are expressed. This property cannot be specified in CTL because of the two nested $*$ operators in the regular subformula of AF.

Fairness properties. Informally, these properties specify the progression of certain concurrent processes in the system, which are possibly antagonistic. In CTRL, fairness properties can be expressed by identifying the infinite sequences of events denoting the proper progression of a certain process, characterizing them using the EF^∞_ρ operator, and requiring their presence in the Kripke structure. The CTRL formula below captures the oscillatory expression patterns of two genes g_1 and g_2 involved in the circadian rhythm:

$$\mathsf{EF}_{\mathsf{true}^*}\mathsf{EF}^\infty_{inc_g_1.\mathsf{true}^*.inc_g_2.\mathsf{true}^*.dec_g_1.\mathsf{true}^*.dec_g_2.\mathsf{true}^*}$$

where the atomic propositions inc_g_1, dec_g_1, inc_g_2, dec_g_2 indicate the increasing and decreasing expression of genes g_1 and g_2, respectively. This property is unexpressible in CTL because of the repeated alternation of inc_g_i and dec_g_i, but it can be stated in LTL using five temporal operators:

$$\begin{aligned}&\neg\mathsf{G}((inc_g_1 \wedge \neg\mathsf{F}inc_g_2) \vee (inc_g_2 \wedge \neg\mathsf{F}dec_g_1)\\ &\vee\ (dec_g_1 \wedge \neg\mathsf{F}dec_g_2) \vee (dec_g_2 \wedge \neg\mathsf{F}inc_g_1))\end{aligned}$$

CTRL was designed such that fairness operators (EF^∞_ρ and AF^∞_ρ) are at the same level as the other temporal operators of the logic. Compared to other logics, such as fair CTL [38], in which fairness constraints are added as side

formulas modifying the interpretation of the temporal operators, we believe that an explicit presence of infinite looping operators in the logic allows a more direct and intuitive description of complex cycles (*e.g.*, matching regular expressions containing nested iteration operators) present in the behaviour of genetic regulatory networks.

2.3. Expressiveness

CTRL is a natural extension of CTL [28], whose main temporal operators can be described using the EF and AF operators of CTRL as follows:

$$\mathsf{E}[\varphi_1 \,\mathsf{U}\, \varphi_2] = \mathsf{EF}_{\varphi_1^*}\, \varphi_2 \qquad\qquad \mathsf{A}[\varphi_1 \,\mathsf{U}\, \varphi_2] = \mathsf{AF}_{\varphi_1^*}\, \varphi_2$$

The until operator U of CTL is not primitive in CTRL; this is a difference w.r.t. other extensions of CTL, such as RCTL [15] and RegCTL [19], which keep the U operator primitive as in the original logic.

CTRL also subsumes LTL [65], because the potential looping operator EF^∞ is able to capture the acceptance condition of Büchi automata. Assuming that the atomic proposition p characterizes the accepting states in a Büchi automaton (represented as a Kripke structure), the formula below expresses the existence of an infinite sequence passing infinitely often through an accepting state:

$$\mathsf{EF}^\infty_{\mathsf{true}^*.p.\mathsf{true}^+}$$

The + operator is necessary in order to avoid empty sequences consisting of a single state satisfying p. Of course, the EF^∞ operator does not allow a direct encoding of LTL operators, but may serve as an intermediate form for LTL model checking; in this respect, EF^∞ is similar to the "never claims" used for specifying properties in the early versions of the SPIN model checker [51].

Thus, CTRL subsumes both CTL and LTL. This subsumption is strict, since these two logics are uncomparable w.r.t. their expressive power (*i.e.*, each one can describe properties unexpressible in the other one) [28]. In fact, the CTRL fragment containing the boolean connectors and the temporal operators EF and EF^∞ can be seen as a state-based variant of PDL-Δ [77]. It was shown that this logic subsumes CTL*, whose operators can be encoded (although not in a succinct way) in PDL-Δ by means of the translation proposed first in [85] and reconsidered later in [30]. This subsumption is strict, because PDL-Δ can characterize sequences that match regular expressions

containing nested iteration operators, such as $(a.b)^*.c$, which cannot be expressed in CTL* [37]. Since CTRL syntactically subsumes (a state-based variant of) PDL-Δ, it follows from the above expressiveness results that CTRL also subsumes CTL*.

As regards other existing extensions of CTL with regular operators, CTRL also subsumes RegCTL, whose U operator indexed by a regular formula can be expressed using the EF operator of CTRL as follows:

$$\mathsf{E}[\varphi_1 \ \mathsf{U}^{\rho} \ \varphi_2] = \mathsf{EF}_{\rho \ \& \ \varphi_1^*}\varphi_2$$

The & operator stands for the intersection of regular formulas; although this operator is not present in CTRL, its occurrence above can be expanded in terms of the regular operators available in CTRL by applying the rules below:

$$\varphi' \ \& \ \varphi^* = \varphi' \ \& \ \varphi \qquad (\rho_1.\rho_2) \ \& \ \varphi^* = (\rho_1 \ \& \ \varphi^*).(\rho_2 \ \& \ \varphi^*)$$
$$(\rho_1|\rho_2) \ \& \ \varphi^* = (\rho_1 \ \& \ \varphi^*)|(\rho_2 \ \& \ \varphi^*) \qquad (\rho_1{}^*) \ \& \ \varphi^* = (\rho_1 \ \& \ \varphi^*)^*$$

The subsumption of RegCTL is strict because the U operator of RegCTL cannot describe an infinite concatenation of intervals satisfying a regular formula ρ, which is specified in CTRL using the $\mathsf{EF}^{\infty}_{\rho}$ operator. In [19] it is shown that RegCTL is more expressive than RCTL [15], the extension of CTL with regular expressions underlying the SUGAR [14] specification language; consequently, CTRL also subsumes RCTL.

3. Translation from CTRL to modal equation systems

Building an efficient model checker for a branching-time temporal logic equipped with regular expressions, such as CTRL, is a complex and time-consuming task. Here we aim at facilitating this task by reusing as much as possible existing verification technology available in the field of concurrent systems, namely the CADP toolbox [44]. A model checker for CTRL can thus be obtained by translating this logic into HMLR [63], one of the input languages accepted by CADP.

This technical section is devoted to the translation of CTRL state formulas into modal equation systems (MESS), which are the state-based counterpart of HMLR. Using such a translation to obtain a model checking procedure for a temporal logic with regular constructs is not common practice, most of the existing procedures for this kind of logics being based on automata [15, 19]. Therefore, we describe this translation in sufficient detail and we illustrate

it with examples. Readers from the systems biology field that are not necessarily interested in the technical details of temporal logic translations can safely skip this section and go directly to Section 4, which provides an overall view of the CTRL model checker and its application to the analysis of genetic regulatory networks.

The translation of a CTRL state formula φ into a MES involves two steps: first the formula is translated into a regular equation system (RES), and then the RES is transformed into a MES. These two steps are purely syntactic, *i.e.*, they do not depend upon the Kripke structure on which the formulas and the equation systems are interpreted. We first define the syntax and semantics of RESs and MESs, and then we detail the two translation steps.

3.1. Regular and modal equation systems

To apply our model checking method, we need to translate CTRL state formulas into an equational representation, which is more suitable than the tree-like representation underlying the syntax definition in Figure 1. To achieve this, we first need to extend the grammar of CTRL state formulas with propositional variables $X \in \mathcal{X}$, which denote sets of states:

$$\varphi ::= X \mid p \mid \ldots$$

Propositional variables are interpreted w.r.t. a Kripke structure K by an environment $\delta : \mathcal{X} \to 2^S$, which is a partial function mapping propositional variables to state sets. The interpretation of state formulas must be extended to handle propositional variables: $[\![\varphi]\!]_K\delta$ denotes the set of states satisfying φ in the context of δ, which must map every variable occurring in φ to a state set. The interpretation of propositional variables is defined as follows: $[\![X]\!]_K\delta = \delta(X)$. The interpretation of the other state formulas defined in Figure 1 remains unchanged, except that an extra parameter δ is added to the interpretation $[\![\ ]\!]$. The translation to MESs ensures that all occurrences of propositional variables in state formulas are positive, *i.e.*, they fall under an even number of negations. This *syntactic monotonicity* condition was proposed initially to ensure the well-definedness of propositional μ-calculus formulas [60].

As intermediate language for translating CTRL state formulas, we use *regular equation systems* (RESs), which are the propositional counterpart of the PDLR (PDL with recursion) specifications introduced in [69]. The syntax and semantics of RESs are defined in Figure 3. Equation blocks B

are sets of fixed point equations having propositional variables $X \in \mathcal{X}$ in the left-hand sides and CTRL state formulas (possibly containing propositional variables) in the right-hand sides. All equations of a block have the same fixed point sign $\sigma \in \{\mu, \nu\}$, where μ and ν denote minimal and maximal fixed points, respectively. The free and bound variables in an equation block list are defined as follows:

$$\begin{array}{ll} fv(\varepsilon) = \emptyset & bv(\varepsilon) = \emptyset \\ fv(B.BL) = (fv(B) \setminus bv(BL)) \cup fv(BL) & bv(B.BL) = bv(B) \cup bv(BL) \\ fv(\{X_i \stackrel{\sigma}{=} \varphi_i\}_{1 \leq i \leq n}) = \bigcup_{i=1}^{n} fv(\varphi_i) & bv(\{X_i \stackrel{\sigma}{=} \varphi_i\}_{1 \leq i \leq n}) = \{X_1, ..., X_n\} \end{array}$$

SYNTAX

$$\begin{array}{rll} R & ::= \langle X, BL \rangle & \text{(regular equation system)} \\ BL & ::= \varepsilon \mid B.BL & \text{(equation block list)} \\ B & ::= \{X_i \stackrel{\sigma}{=} \varphi_i\}_{1 \leq i \leq n} & \text{(equation block)} \end{array}$$

SEMANTICS

$$\begin{array}{rcl} [\![\langle X, BL \rangle]\!]_K & = & ([\![BL]\!]_K)(X) \\ [\![\varepsilon]\!]_K \delta & = & [\,] \\ [\![B.BL]\!]_K \delta & = & [\![B]\!]_K(\delta \oslash [\![BL]\!]_K \delta) \oslash [\![BL]\!]_K \delta \\ [\![\{X_i \stackrel{\sigma}{=} \varphi_i\}_{1 \leq i \leq n}]\!]_K \delta & = & [(\sigma\Phi_\delta)_1/X_1, ..., (\sigma\Phi_\delta)_n/X_n] \end{array}$$

$\Phi_\delta : (2^S)^n \to (2^S)^n$, $\Phi_\delta(U_1, ..., U_n) = \langle [\![\varphi_i]\!]_K(\delta \oslash [U_1/X_1, ..., U_n/X_n]) \rangle_{1 \leq i \leq n}$

Figure 3: Syntax and semantics of regular equation systems

The set $fv(\varphi_i)$ contains all propositional variables occurring in φ_i. A block list BL is *closed* if $fv(BL) = \emptyset$. We consider that all nonempty block lists $B.BL$ satisfy the following conditions: $bv(B) \cap bv(BL) = \emptyset$ (*normal form*) and $fv(B) \subseteq bv(B) \cup bv(BL)$ (*alternation-free*). In a block list $B.BL$, block B depends upon another block B' of BL if $fv(B) \cap bv(B') \neq \emptyset$, *i.e.*, B contains a free variable bound in B'. The alternation-free condition means that there are no cyclic dependencies between equation blocks, and block B depends only upon the blocks of BL, placed at his right in the list $B.BL$. In a RES $R = \langle X, BL \rangle$, BL is assumed to be nonempty and closed. X is called the *main variable* and must be bound in the first block of BL.

The interpretation of a RES $R = \langle X, BL \rangle$ on a Kripke structure $K = \langle S, P, L, T, s_0 \rangle$ is the value of variable X as obtained by solving the block

list BL. The interpretation $[\![BL]\!]_K\delta$ of a block list in the context of an environment δ is another environment assigning state sets to all variables bound in BL. Since the blocks of BL depend upon each other from left to right, the interpretation of BL can be defined inductively, by solving the blocks from right to left. The notation $\delta \oslash [U_1/X_1, ..., U_n/X_n]$ stands for the extension of δ with $[U_1/X_1, ..., U_n/X_n]$, *i.e.*, an environment identical to δ except for variables $X_1, ..., X_n$, which are mapped to the state sets $U_1, ..., U_n$, respectively. The empty environment is noted [ ]. The interpretation of an equation block B is the environment mapping the variables bound in B to the state sets given by the corresponding fixed point of the functional associated to the block. When BL is closed, the δ environment is omitted. The state formulas in the right-hand sides of equations are assumed to be syntactically monotonic, which according to Tarski's theorem [78] ensures the well-definedness of the functionals associated to blocks.

A *modal equation system* (MES) $M = \langle X, BL\rangle$ is a RES where all CTRL temporal operators occurring in the right-hand sides of equations contain only atomic regular formulas, *i.e.*, without any regular operator ('.', '|', '*'). MESs are the propositional counterpart of the HMLR (HML with recursion) specifications, proposed in [63] as an equational form of the modal μ-calculus.

3.2. Translation to regular equation systems

The translation of a CTRL state formula φ into a RES is defined by the syntactic function $t(\varphi) = \langle t_X(\varphi), t_{BL}(\varphi)\rangle$ given in Figure 4. The two components $t_X(\varphi)$ and $t_{BL}(\varphi)$ denote the main variable and the equation block list produced by $t(\varphi)$, respectively. For each translation rule, X denotes a "fresh" propositional variable, different from all the other variables contained in φ and in $t(\varphi)$. The notation $BL_1; BL_2$ indicates the concatenation of two equation block lists BL_1, BL_2 and is defined inductively as follows: $\varepsilon; BL_2 = BL_2$, and $(B.BL_1); BL_2 = B.(BL_1; BL_2)$.

For simplicity, in the translation of propositional constants we omitted the empty block list, *i.e.*, we wrote $\{X \stackrel{\mu}{=} p\}$ instead of $\{X \stackrel{\mu}{=} p\}.\varepsilon$. If φ is closed, then the block list produced by the translation is also closed, *i.e.*, $bv(t_{BL}(\varphi)) = \emptyset$. The translation given in Figure 4 preserves the interpretation of formulas, as stated by the proposition below.

Proposition 1 (Translation from CTRL to RESs). *Let K be a Kripke structure and φ a state formula of* CTRL. *Then:*

$$[\![\varphi]\!]_K\delta = [\![t(\varphi)]\!]_K\delta$$

for any propositional environment δ.

$$
\begin{aligned}
t(p) &= \langle X, \{X \stackrel{\mu}{=} p\}\rangle \\
t(\varphi_1 \vee \varphi_2) &= \langle X, \{X \stackrel{\mu}{=} t_X(\varphi_1) \vee t_X(\varphi_2)\}.(t_{BL}(\varphi_1); t_{BL}(\varphi_2))\rangle \\
t(\varphi_1 \wedge \varphi_2) &= \langle X, \{X \stackrel{\mu}{=} t_X(\varphi_1) \wedge t_X(\varphi_2)\}.(t_{BL}(\varphi_1); t_{BL}(\varphi_2))\rangle \\
t(\mathsf{EF}_\rho\varphi) &= \langle X, \{X \stackrel{\mu}{=} \mathsf{EF}_\rho t_X(\varphi)\}.t_{BL}(\varphi)\rangle \\
t(\mathsf{AF}_\rho\varphi) &= \langle X, \{X \stackrel{\mu}{=} \mathsf{AF}_\rho t_X(\varphi)\}.t_{BL}(\varphi)\rangle \\
t(\mathsf{EG}_\rho\varphi) &= \langle X, \{X \stackrel{\nu}{=} \mathsf{EG}_\rho t_X(\varphi)\}.t_{BL}(\varphi)\rangle \\
t(\mathsf{AG}_\rho\varphi) &= \langle X, \{X \stackrel{\nu}{=} \mathsf{AG}_\rho t_X(\varphi)\}.t_{BL}(\varphi)\rangle \\
t(\mathsf{EF}^\infty_\rho) &= \langle X, \{X \stackrel{\nu}{=} \mathsf{EF}_\rho X\}\rangle \\
t(\mathsf{AF}^\infty_\rho) &= \langle X, \{X \stackrel{\nu}{=} \mathsf{AF}_\rho X\}\rangle \\
t(\mathsf{EG}^\dashv_\rho) &= \langle X, \{X \stackrel{\mu}{=} \mathsf{EG}_\rho X\}\rangle \\
t(\mathsf{AG}^\dashv_\rho) &= \langle X, \{X \stackrel{\mu}{=} \mathsf{AG}_\rho X\}\rangle
\end{aligned}
$$

Figure 4: Translation of CTRL state formulas into RESs

To illustrate the translation of CTRL formulas into RESs, we consider the *bistability property* [80, 35], which specifies that after an initial state, two different equilibrium states can be potentially reached. This branching-time property can be expressed in CTRL as follows:

$$\mathsf{AG}_{\mathsf{true}^*.init}(\mathsf{EF}_{\mathsf{true}^*}\, eql_1 \wedge \mathsf{EF}_{\mathsf{true}^*}\, eql_2)$$

where the atomic propositions *init*, eql_1, and eql_2 denote the initial state and the two equilibrium states, respectively. By applying the translation defined in Figure 4 to this formula, we obtain the RES below:

$$
\begin{aligned}
&\langle X, \{X \stackrel{\nu}{=} \mathsf{AG}_{\mathsf{true}^*.init} Y\}.\{Y \stackrel{\mu}{=} Z_1 \wedge Z_2\}. \\
&\quad \{Z_1 \stackrel{\mu}{=} \mathsf{EF}_{\mathsf{true}^*} U_1\}.\{U_1 \stackrel{\mu}{=} eql_1\}.\{Z_2 \stackrel{\mu}{=} \mathsf{EF}_{\mathsf{true}^*} U_2\}.\{U_2 \stackrel{\mu}{=} eql_2\}.\varepsilon\rangle
\end{aligned}
$$

The ';' operator produced by translating $\mathsf{EF}_{\mathsf{true}^*}\, eql_1 \wedge \mathsf{EF}_{\mathsf{true}^*}\, eql_2$ was expanded in terms of the '.' operator using the definition of ';'.

The size (number of variables and operators) of the RES $t(\varphi)$ produced by the translation is linear in the size (number of operators) of the formula φ, because every rule in Figure 4 creates, for each operator present in φ, one block containing a single equation with one operator in its right-hand side.

For simplicity, the translation of a state formula φ given in Figure 4 does not take care of the state subformulas ψ that may occur inside the regular formulas ρ. However, these subformulas must also be translated into RESs in order to be evaluated on a Kripke structure K by the model checking procedure. This is done by applying the translation recursively on every subformula ψ of a regular formula ρ, yielding an additional RES $t(\psi) = \langle t_X(\psi), t_{BL}(\psi)\rangle$. In practice, the block list $t_{BL}(\psi)$ of each additional RES $t(\psi)$ is concatenated to the block list $t_{BL}(\varphi)$ of the RES $t(\varphi)$, and the main variable $t_X(\psi)$ replaces the occurrence of the corresponding subformula ψ, as illustrated by the formula $\mathsf{EF}_{(\mathsf{AG}_{\mathsf{true}^*}p)^*}q$, whose translation yields the RES $\langle X, \{X \stackrel{\mu}{=} \mathsf{EF}_{Y^*}Z\}.\{Z \stackrel{\mu}{=} q\}.\{Y \stackrel{\nu}{=} \mathsf{AG}_{\mathsf{true}^*}U\}.\{U \stackrel{\nu}{=} p\}.\varepsilon\rangle$.

However, in order to simplify notations, we can exploit the fact that the RESs produced by translating the subformulas ψ are closed, and hence their main variables can be evaluated independently from the RES $t(\varphi)$. This allows to safely replace each subformula ψ by a "fresh" atomic proposition p_ψ, whose interpretation on K is obtained by evaluating the main variable $t_X(\psi)$ of the RES $t(\psi)$. On the example above, the RES becomes $\langle X, \{X \stackrel{\mu}{=} \mathsf{EF}_{r^*}Z\}.\{Z \stackrel{\mu}{=} q\}.\varepsilon\rangle$, where r has the same interpretation as the variable Y of the additional RES $\langle Y, \{Y \stackrel{\nu}{=} \mathsf{AG}_{\mathsf{true}^*}U\}.\{U \stackrel{\nu}{=} p\}.\varepsilon\rangle$. Therefore, in the sequel we will restrict ourselves to RESs in which the regular formulas occurring in the right-hand sides of equations are built only upon atomic propositions.

3.3. Translation to modal equation systems

Let $B = \{X_i \stackrel{\sigma}{=} \varphi_i\}_{1\leq i\leq n}$ be an equation block. An equation block $\{X_n \stackrel{\sigma}{=} \psi_n, Y_j \stackrel{\sigma}{=} \psi_j\}_{n<j\leq m}$ is *suitable* for the substitution of equation $X_n \stackrel{\sigma}{=} \varphi_n$ if $fv(\psi_n) \cup \bigcup_{j=n+1}^{m} fv(\psi_j) = fv(\varphi_n)$ and $\bigcup_{i=1}^{n} fv(\varphi_i) \cap \{Y_{n+1}, ..., Y_m\} = \emptyset$. The notation $\{X_i \stackrel{\sigma}{=} \varphi_i\}_{1\leq i\leq n}[X_n \stackrel{\sigma}{=} \varphi_n := X_n \stackrel{\sigma}{=} \psi_n, Y_j \stackrel{\sigma}{=} \psi_j\}_{n<j\leq m}]$ represents the syntactic substitution of the equation $X_n \stackrel{\sigma}{=} \varphi_n$ by the equations $\{X_n \stackrel{\sigma}{=} \psi_n, Y_j \stackrel{\sigma}{=} \psi_j\}_{n<j\leq m}$ in B. This definition of substitution, which allows to replace only the last equation of a block, is general enough: since all equations of a block have the same fixed point sign, their order does not influence the values of the variables defined in the block, and therefore any equation of the block can be substituted by bringing it in the last position.

The translation of a RES equation block into a corresponding MES equation block is performed by repeatedly applying various transformations, most of them being substitutions of equations.

3.3.1. Operators EF_ρ *and* AG_ρ

In order to translate the equation blocks of the form $\{X \stackrel{\mu}{=} \mathsf{EF}_\rho Y\}$ and $\{X \stackrel{\nu}{=} \mathsf{AG}_\rho Y\}$ into MEss, we eliminate the regular expressions ρ by repeatedly applying appropriate substitutions. Each equation containing an EF_ρ or AG_ρ operator in its right-hand side is substituted with a suitable equation block containing simpler regular formulas, as defined in Figure 5 (Z and U are "fresh" propositional variables). The application of any substitution given in Figure 5 preserves the interpretation of equation blocks, as stated by the proposition below.

EQUATION	SUBSTITUTION BLOCK
$X \stackrel{\mu}{=} \mathsf{EF}_{\rho_1 . \rho_2} Y$	$\{X \stackrel{\mu}{=} \mathsf{EF}_{\rho_1} Z, Z \stackrel{\mu}{=} \mathsf{EF}_{\rho_2} Y\}$
$X \stackrel{\mu}{=} \mathsf{EF}_{\rho_1 \mid \rho_2} Y$	$\{X \stackrel{\mu}{=} Z \vee U, Z \stackrel{\mu}{=} \mathsf{EF}_{\rho_1} Y, U \stackrel{\mu}{=} \mathsf{EF}_{\rho_2} Y\}$
$X \stackrel{\mu}{=} \mathsf{EF}_{\rho^*} Y$	$\{X \stackrel{\mu}{=} Y \vee Z, Z \stackrel{\mu}{=} \mathsf{EF}_{\rho} X\}$
$X \stackrel{\nu}{=} \mathsf{AG}_{\rho_1 . \rho_2} Y$	$\{X \stackrel{\nu}{=} \mathsf{AG}_{\rho_1} Z, Z \stackrel{\nu}{=} \mathsf{AG}_{\rho_2} Y\}$
$X \stackrel{\nu}{=} \mathsf{AG}_{\rho_1 \mid \rho_2} Y$	$\{X \stackrel{\nu}{=} Z \wedge U, Z \stackrel{\nu}{=} \mathsf{AG}_{\rho_1} Y, U \stackrel{\nu}{=} \mathsf{AG}_{\rho_2} Y\}$
$X \stackrel{\nu}{=} \mathsf{AG}_{\rho^*} Y$	$\{X \stackrel{\nu}{=} Y \wedge Z, Z \stackrel{\nu}{=} \mathsf{AG}_{\rho} X\}$

Figure 5: Substitutions for the EF_ρ and AG_ρ operators

Proposition 2 (Substitution of EF and AG). *Let K be a Kripke structure and $B_1 = \{X_i \stackrel{\mu}{=} \varphi_i\}_{1 \leq i \leq n}$, $B_2 = \{X_i \stackrel{\nu}{=} \varphi_i\}_{1 \leq i \leq n}$ be two equation blocks. Then, for any propositional environment δ, the interpretation of B_1 (resp. B_2) w.r.t. δ does not change when a substitution given in the upper part (resp. the lower part) of Figure 5 is applied.*

By repeatedly applying these substitutions, all occurrences of regular operators in the right-hand sides of the equations can be eliminated. For the RES of the bistability property, this translation yields the following MES:

$$\begin{array}{l} \langle X, \{X \stackrel{\nu}{=} Y_1 \wedge Y_2,\ Y_1 \stackrel{\nu}{=} \mathsf{AG}_{init} Y,\ Y_2 \stackrel{\nu}{=} \mathsf{AG}_{\mathsf{true}} X\}.\{Y \stackrel{\mu}{=} Z_1 \wedge Z_2\}. \\ \quad \{Z_1 \stackrel{\mu}{=} U_1 \vee Z_3,\ Z_3 \stackrel{\mu}{=} \mathsf{EF}_{\mathsf{true}} Z_1\}.\{U_1 \stackrel{\mu}{=} eql_1\}. \\ \quad \{Z_2 \stackrel{\mu}{=} U_2 \vee Z_4,\ Z_4 \stackrel{\mu}{=} \mathsf{EF}_{\mathsf{true}} Z_2\}.\{U_2 \stackrel{\mu}{=} eql_2\}.\varepsilon\rangle \end{array}$$

The equation block $\{X \stackrel{\nu}{=} \mathsf{AG}_{\mathsf{true}^* . init} Y\}$ was translated by successively applying the first and the third substitutions in the lower part of Figure 5.

The size of the MES equation block resulting from the translation of a RES equation block B of the form $\{X \stackrel{\mu}{=} \mathsf{EF}_\rho Y\}$ (resp. $\{X \stackrel{\nu}{=} \mathsf{AG}_\rho Y\}$) remains linear w.r.t. the size of B (and hence linear w.r.t. the size of the initial CTRL formula φ), since each substitution in Figure 5 replaces a regular operator by at most two variables and two temporal operators EF (resp. AG).

3.3.2. Operators AF_ρ and EG_ρ

The translation of the equation blocks $\{X \stackrel{\mu}{=} \mathsf{AF}_\rho Y\}$ and $\{X \stackrel{\nu}{=} \mathsf{EG}_\rho Y\}$ into MESs is more complicated than the translation of their EF_ρ and AG_ρ counterparts, because the substitutions given in Figure 5 to eliminate the regular expressions ρ are no longer valid for the AF_ρ and EG_ρ operators. We consider below only blocks of the form $\{X \stackrel{\mu}{=} \mathsf{AF}_\rho Y\}$, the processing of their EG_ρ counterparts being dual.

The translation of the $\{X \stackrel{\mu}{=} \mathsf{AF}_\rho Y\}$ equation blocks into MESs consists of three steps. First, the RES is temporarily transformed in *potentiality* form $\{X \stackrel{\mu}{=} \mathsf{EF}_\rho Y\}$ and subsequently translated into a potentiality MES by eliminating the regular expression ρ using the substitutions given in Section 3.3.1. Then, the resulting MES is transformed in *guarded* form, by eliminating all unguarded (*i.e.*, not preceded by a temporal operator) occurrences of variables in the right-hand sides of equations. Finally, the guarded MES is *determinized*, by replacing all occurrences of EF operators in the right-hand sides of equations by appropriate occurrences of AF operators in order to retrieve the interpretation of the initial equation block $\{X \stackrel{\mu}{=} \mathsf{AF}_\rho Y\}$. We will illustrate each step of the translation on the following equation block:

$$\{X \stackrel{\mu}{=} \mathsf{AF}_{(q|p^*)^*.(qr^*).(p^*|q^*)} Y\}.$$

Translation to potentiality form. The difficulty of translating an equation block $\{X \stackrel{\mu}{=} \mathsf{AF}_\rho Y\}$ into a MES stems from the fact that *all* transition sequences going out of a state have to satisfy ρ before reaching a state satisfying Y, whereas the substitutions in Figure 5 allow to eliminate ρ on individual sequences only. To avoid this difficulty, we switch temporarily to the potentiality form $\{X \stackrel{\mu}{=} \mathsf{EF}_\rho Y\}$, we eliminate ρ by applying the substitutions, and we continue working with the resulting potentiality MES, which characterizes the existence of individual sequences satisfying ρ. The size of this MES is linear w.r.t. the size of the initial block $\{X \stackrel{\mu}{=} \mathsf{AF}_\rho Y\}$, as stated in Section 3.3.1. Figure 6 shows the potentiality MES obtained from the equation block above by switching to potentiality form and applying the substitutions in Figure 5 (all equations have the sign μ, omitted for simplicity).

$X = \mathsf{EF}_{(q\|p^*)^*} Z_1$	$X = Z_3 \vee Z_1$			$X = Z_3 \vee Z_1$
	$Z_3 = \mathsf{EF}_{q\|p^*} X$	$Z_3 = Z_4 \vee Z_5$		$Z_3 = Z_4 \vee Z_5$
		$Z_5 = \mathsf{EF}_q X$		$Z_5 = \mathsf{EF}_q X$
		$Z_4 = \mathsf{EF}_{p^*} X$	$Z_4 = Z_6 \vee X$	$Z_4 = Z_6 \vee X$
			$Z_6 = \mathsf{EF}_p Z_4$	$Z_6 = \mathsf{EF}_p Z_4$
$Z_1 = \mathsf{EF}_{qr^*} Z_2$	$Z_1 = \mathsf{EF}_q Z_7$			$Z_1 = \mathsf{EF}_q Z_7$
	$Z_7 = \mathsf{EF}_{r^*} Z_2$	$Z_7 = Z_8 \vee Z_2$		$Z_7 = Z_8 \vee Z_2$
		$Z_8 = \mathsf{EF}_r Z_7$		$Z_8 = \mathsf{EF}_r Z_7$
$Z_2 = \mathsf{EF}_{p^*\|q^*} Y$	$Z_2 = Z_9 \vee Z_{10}$			$Z_2 = Z_9 \vee Z_{10}$
	$Z_9 = \mathsf{EF}_{p^*} Y$	$Z_9 = Z_{11} \vee Y$		$Z_9 = Z_{11} \vee Y$
		$Z_{11} = \mathsf{EF}_p Z_9$		$Z_{11} = \mathsf{EF}_p Z_9$
	$Z_{10} = \mathsf{EF}_{q^*} Y$	$Z_{10} = Z_{12} \vee Y$		$Z_{10} = Z_{12} \vee Y$
		$Z_{12} = \mathsf{EF}_q Z_{10}$		$Z_{12} = \mathsf{EF}_q Z_{10}$

Figure 6: Translation of $\{X \stackrel{\mu}{=} \mathsf{AF}_{(q|p^*)^*.(qr^*).(p^*|q^*)} Y\}$ to a potentiality Mes

The right-hand sides of the equations of the potentiality Mes may contain unguarded occurrences of propositional variables (*i.e.*, not preceded by any EF operator), such as variable Z_1 in the equation $X = Z_3 \vee Z_1$. These occurrences will be eliminated in the next step of the translation.

Translation to guarded form. The translation of a potentiality Mes to guarded form eliminates all unguarded occurrences of variables in the right-hand sides of equations using the lemma below.

Lemma 1 (Absorption). *Let K be a Kripke structure and $B = \{X_i \stackrel{\mu}{=} \varphi_i\}_{1 \leq i \leq n}$ be an equation block such that $\varphi_n = X_n \vee \varphi$ and $X_n \notin fv(\varphi)$. Then:*

$$[\![\{X_i \stackrel{\mu}{=} \varphi_i\}_{1 \leq i \leq n}[X_n \stackrel{\mu}{=} X_n \vee \varphi := X_n \stackrel{\mu}{=} \varphi]]\!]_K \delta = [\![\{X_i \stackrel{\mu}{=} \varphi_i\}_{1 \leq i \leq n}]\!]_K \delta$$

for any propositional environment δ.

The equations of a potentiality Mes, produced by the rules in Figure 5, have two possible forms: either unguarded (*i.e.*, containing disjunctions of variables in their right-hand side), or guarded (*i.e.*, containing a single occurrence of an EF operator in their right-hand side). The elimination of unguarded occurrences of variables is carried out by Algorithm 1. The first loop of the algorithm applies the absorption lemma and the idempotency of disjunction on each unguarded equation defining a variable X in order to

eliminate the unguarded occurrences of X, and afterwards expands inline all unguarded occurrences of X in all the other equations of the MES. After executing the first loop on the potentiality MES given in Figure 6, we obtain the MES shown in Figure 7. Upon termination of the first loop, the formulas in the right-hand sides of equations may contain only unguarded occurrences of Y and of variables X defined by guarded equations of the MES. The second loop of the algorithm expands inline those variables, thus eliminating all unguarded occurrences except those of Y. The result of applying the second loop on the MES in Figure 7 yields the MES shown in Figure 8.

Algorithm 1 Translation of a potentiality MES to guarded form

for all unguarded equations $X \stackrel{\mu}{=} \bigvee_j X_j$ **do**
  Eliminate X among X_j by applying the absorption lemma
  for all unguarded occurrences of X in the rsh of other equations **do**
    Substitute X by $\bigvee_j X_j$
  end for
end for
for all guarded equations $X \stackrel{\mu}{=} \mathsf{EF}_p X_j$ **do**
  Substitute X by $\mathsf{EF}_p X_j$ in all unguarded equations
end for

Initial list of unguarded eqns.	1^{st} loop of Algorithm 1	
	Var.	Updated equations
$X = Z_3 \vee Z_1$	X:	$Z_4 = Z_6 \vee Z_3 \vee Z_1$
$Z_2 = Z_9 \vee Z_{10}$	Z_2:	$Z_7 = Z_8 \vee Z_9 \vee Z_{10}$
$Z_3 = Z_4 \vee Z_5$	Z_3:	$X = Z_4 \vee Z_5 \vee Z_1$
$Z_4 = Z_6 \vee X$		$Z_4 = Z_6 \vee \cancel{Z_4} \vee Z_5 \vee Z_1$
$Z_7 = Z_8 \vee Z_2$	Z_4:	$X = Z_6 \vee Z_5 \vee Z_1 \vee \cancel{Z_5} \vee \cancel{Z_1}$
$Z_9 = Z_{11} \vee Y$		$Z_3 = Z_1 \vee Z_5 \vee Z_6 \vee \cancel{Z_5}$
$Z_{10} = Z_{12} \vee Y$	Z_9:	$Z_2 = Z_{11} \vee Y \vee Z_{10}$
		$Z_7 = Z_8 \vee Z_{11} \vee Y \vee Z_{10}$
	Z_{10}:	$Z_2 = Z_{11} \vee Y \vee Z_{12} \vee \cancel{Y}$
		$Z_7 = Z_8 \vee Z_{11} \vee Y \vee Z_{12} \vee \cancel{Y}$

Figure 7: Translation of a potentiality MES to guarded form (1^{st} part)

The guarded MESs obtained by applying Algorithm 1 can be further simplified by eliminating duplicate and unreachable equations. In the MES

Equations after the 1^{st} loop	2^{nd} loop of Algorithm 1	
	Var.	Updated equations
$X = Z_6 \vee Z_5 \vee Z_1$	Z_1:	$X = Z_6 \vee Z_5 \vee \mathsf{EF}_q Z_7$
$Z_2 = Z_{11} \vee Y \vee Z_{12}$		$Z_3 = \mathsf{EF}_q Z_7 \vee Z_5 \vee Z_6$
$Z_3 = Z_1 \vee Z_5 \vee Z_6$		$Z_4 = Z_6 \vee Z_5 \vee \mathsf{EF}_q Z_7$
$Z_4 = Z_6 \vee Z_5 \vee Z_1$	Z_5:	$X = Z_6 \vee \mathsf{EF}_q X \vee \mathsf{EF}_q Z_7$
$Z_7 = Z_8 \vee Z_{11} \vee Y \vee Z_{12}$		$Z_3 = \mathsf{EF}_q Z_7 \vee \mathsf{EF}_q X \vee Z_6$
$Z_9 = Z_{11} \vee Y$		$Z_4 = Z_6 \vee \mathsf{EF}_q X \vee \mathsf{EF}_q Z_7$
$Z_{10} = Z_{12} \vee Y$	Z_6:	$X = \mathsf{EF}_p Z_4 \vee \mathsf{EF}_q X \vee \mathsf{EF}_q Z_7$
$Z_1 = \mathsf{EF}_q Z_7$		$Z_3 = \mathsf{EF}_q Z_7 \vee \mathsf{EF}_q X \vee \mathsf{EF}_p Z_4$
$Z_5 = \mathsf{EF}_q X$		$Z_4 = \mathsf{EF}_p Z_4 \vee \mathsf{EF}_q X \vee \mathsf{EF}_q Z_7$
$Z_6 = \mathsf{EF}_p Z_4$	Z_8:	$Z_7 = \mathsf{EF}_r Z_7 \vee Z_{11} \vee Y \vee Z_{12}$
$Z_8 = \mathsf{EF}_r Z_7$	Z_{11}:	$Z_2 = \mathsf{EF}_p Z_9 \vee Y \vee Z_{12}$
$Z_{11} = \mathsf{EF}_p Z_9$		$Z_7 = \mathsf{EF}_r Z_7 \vee \mathsf{EF}_p Z_9 \vee Y \vee Z_{12}$
$Z_{12} = \mathsf{EF}_q Z_{10}$		$Z_9 = \mathsf{EF}_p Z_9 \vee Y$
	Z_{12}:	$Z_2 = \mathsf{EF}_p Z_9 \vee Y \vee \mathsf{EF}_q Z_{10}$
		$Z_7 = \mathsf{EF}_r Z_7 \vee \mathsf{EF}_p Z_9 \vee Y \vee \mathsf{EF}_q Z_{10}$
		$Z_{10} = \mathsf{EF}_q Z_{10} \vee Y$

Figure 8: Translation of a potentiality MES to guarded form (2^{nd} part)

$$\left\{\begin{array}{ll} X & = \mathsf{EF}_p X \vee \mathsf{EF}_q X \vee \mathsf{EF}_q Z_7 \\ Z_7 & = \mathsf{EF}_r Z_7 \vee \mathsf{EF}_p Z_9 \vee \mathsf{EF}_q Z_{10} \vee Y \\ Z_{10} & = \mathsf{EF}_q Z_{10} \vee Y \\ Z_9 & = \mathsf{EF}_p Z_9 \vee Y \end{array}\right\} \quad \left\{\begin{array}{l} X_1 = \mathsf{EF}_p X_1 \vee \mathsf{EF}_q X_1 \vee \mathsf{EF}_q X_2 \\ X_2 = \mathsf{EF}_p X_4 \vee \mathsf{EF}_q X_3 \vee \mathsf{EF}_r X_2 \vee Y \\ X_3 = \mathsf{EF}_q X_3 \vee Y \\ X_4 = \mathsf{EF}_p X_4 \vee Y \end{array}\right\}$$

Figure 9: Guarded potentiality MES after simplifications (left) and renaming (right)

shown in Figure 8, the equations defining X, Z_3 and Z_4 have identical right-hand sides, and therefore variables Z_4 and Z_3 can be replaced by X and their equations deleted. Also, some of the variables will no longer be referenced after these substitutions, and therefore their equations can be safely removed. Finally, variables can be renamed in order to have a proper numbering, leading to the MES shown in Figure 9. This guarded MES is equivalent to $\{X \stackrel{\mu}{=} \mathsf{EF}_{(q|p^*)^*.(qr^*).(p^*|q^*)} Y\}$, the potentiality form of our running example. Intuitively, each variable defined by this MES denotes the suffix of a transition sequence in the Kripke structure satisfying the regular formula indexing the EF operator. In this respect, guarded potentiality MESs are similar to

the equation systems defining the derivatives of regular expressions [20].

The guarded potentiality MESs produced by applying Algorithm 1 have at most the same number of variables as the original MESs, but may present in the worst-case a quadratic increase in the number of operators. However, we observed in practice that the number of variables in the guarded MESs is much smaller than in the original MESs (thanks to elimination of redundant equations) and the number of operators remains close to linear w.r.t. the original MESs, and hence w.r.t. the size of the initial CTRL formula.

Determinization. The last step of the translation consists in determinizing the guarded potentiality MES obtained so far in order to obtain a MES with the same meaning as the initial RES $\{X \stackrel{\mu}{=} \mathsf{AF}_\rho Y\}$. Consider the following potentiality MES in guarded form:

$$\left\{ X_i \stackrel{\mu}{=} \bigvee_{j=1}^{n} (h_{ij} \wedge \mathsf{EF}_{p_{ij}} X_j) \vee (h_i \wedge Y) \right\}_{1 \leq i \leq n}$$

where $h_{ij}, h_i \in \mathbf{Bool}$ and $p_{ij} \in P$ for all $1 \leq i, j \leq n$. The coefficients h_{ij} and h_i allow to simplify notations: only the terms $\mathsf{EF}_{p_{ij}} X_j$ with their coefficients h_{ij} equal to true (and similarly for the unguarded occurrences of Y with their h_i equal to true) are present in the right-hand sides of equations. An equation defining variable X_i is said to have the index i. Note that the translation to guarded potentiality form may produce equations containing guarded occurrences of Y, *e.g.*, formulas $\mathsf{EF}_p Y$ in their right-hand sides; in this case, bringing the MES to the form above requires to introduce an extra equation $X_{n+1} \stackrel{\mu}{=} Y$ and to replace by X_{n+1} all guarded occurrences of Y (but not its unguarded occurrences). The *determinized* MES corresponding to the guarded potentiality MES above is defined as follows:

$$\left\{ X_I \stackrel{\mu}{=} \bigvee_{\emptyset \subset Q \subseteq prop(I)} \mathsf{AF}_Q X_{vars(Q,I)} \vee (h(I) \wedge Y) \right\}_{I \subseteq [1,n]}$$

where:

- $prop(I) \stackrel{d}{=} \{p_{ij} \mid i \in I \wedge j \in [1,n] \wedge h_{ij}\}$ is the set of atomic propositions occurring as subscripts of EF operators in the equations of the guarded potentiality MES having their index in the set I.

- $vars(Q, I) \stackrel{d}{=} \{j \in [1, n] \mid \exists i \in I.(h_{ij} \wedge p_{ij} \in Q)\}$ is the set of indexes of propositional variables which occur in the right-hand side of some equation having its index in the set I and whose corresponding EF operator is subscripted by some atomic proposition contained in the set Q.

- $h(I) \stackrel{d}{=} \exists i \in I.h_i$ is equal to true iff Y occurs unguarded in some equation having its index in the set I.

In the AF operators of the determinized MES, the subscript Q stands for the conjunction of all the atomic propositions contained in the set Q.

The determinization restores the meaning of the initial equation block $\{X \stackrel{\mu}{=} \mathsf{AF}_\rho Y\}$, as stated by the proposition below.

Proposition 3 (Determinization correctness). *Let K be a Kripke structure, $R = \{X_1 \stackrel{\mu}{=} \mathsf{AF}_\rho Y\}$ an equation block, and M the* MES *obtained from R after translation in guarded potentiality form and determinization. Then:*

$$([\![M]\!]_K \delta)(X_{\{1\}}) = ([\![R]\!]_K \delta)(X_1)$$

for any propositional environment δ.

Figure 10 shows the determinized version of the guarded potentiality MES produced by the previous translation phases from the equation block $\{X \stackrel{\mu}{=} \mathsf{AF}_{(q|p^*)^*.(qr^*).(p^*|q^*)} Y\}$. For conciseness, we represent index sets just by concatenating their elements, *e.g.*, the set $\{1, 2, 3\}$ is denoted by 123. We observe that this MES can be simplified by eliminating duplicate equations (*e.g.*, the equations defining variables X_{12}, X_{123}, X_{124} and those defining $X_2, X_{23}, X_{24}, X_{234}$) and by absorbing certain operands using the identity $\mathsf{AF}_p X_I \vee \mathsf{AF}_{pq} X_I = \mathsf{AF}_p X_I$, yielding the MES on the left of Figure 11. Finally, the right-hand side formulas of some equations may occur as subformulas in other equations and can therefore be replaced by their corresponding left-hand side variables, leading to the final determinized MES shown on the right of Figure 11. In practice, these simplifications can be carried out incrementally as the equations are generated, avoiding the complete construction of the determinized MES prior to simplification. Moreover, sometimes it is possible to determine statically whether certain atomic propositions are mutually exclusive, which allows to remove the AF operators whose index subformulas contain those propositions together.

$$\left\{\begin{array}{ll}
X_1 & \stackrel{\mu}{=} \mathsf{AF}_p X_1 \vee \mathsf{AF}_q X_{12} \vee \mathsf{AF}_{pq} X_{12} \\
X_{12} & \stackrel{\mu}{=} \mathsf{AF}_p X_{14} \vee \mathsf{AF}_q X_{123} \vee \mathsf{AF}_r X_2 \vee \mathsf{AF}_{pq} X_{1234} \vee \mathsf{AF}_{pr} X_{124} \vee \mathsf{AF}_{qr} X_{123} \vee \mathsf{AF}_{pqr} X_{1234} \vee Y \\
X_{123} & \stackrel{\mu}{=} \mathsf{AF}_p X_{14} \vee \mathsf{AF}_q X_{123} \vee \mathsf{AF}_r X_2 \vee \mathsf{AF}_{pq} X_{1234} \vee \mathsf{AF}_{pr} X_{124} \vee \mathsf{AF}_{qr} X_{123} \vee \mathsf{AF}_{pqr} X_{1234} \vee Y \\
X_{124} & \stackrel{\mu}{=} \mathsf{AF}_p X_{14} \vee \mathsf{AF}_q X_{123} \vee \mathsf{AF}_r X_2 \vee \mathsf{AF}_{pq} X_{1234} \vee \mathsf{AF}_{pr} X_{124} \vee \mathsf{AF}_{qr} X_{123} \vee \mathsf{AF}_{pqr} X_{1234} \vee Y \\
X_{14} & \stackrel{\mu}{=} \mathsf{AF}_p X_{14} \vee \mathsf{AF}_q X_{12} \vee \mathsf{AF}_{pq} X_{124} \vee Y \\
X_{1234} & \stackrel{\mu}{=} \mathsf{AF}_p X_{14} \vee \mathsf{AF}_q X_{123} \vee \mathsf{AF}_r X_2 \vee \mathsf{AF}_{pq} X_{1234} \vee \mathsf{AF}_{pr} X_{124} \vee \mathsf{AF}_{qr} X_{123} \vee \mathsf{AF}_{pqr} X_{1234} \vee Y \\
X_2 & \stackrel{\mu}{=} \mathsf{AF}_p X_4 \vee \mathsf{AF}_q X_3 \vee \mathsf{AF}_r X_2 \vee \mathsf{AF}_{pq} X_{34} \vee \mathsf{AF}_{pr} X_{24} \vee \mathsf{AF}_{qr} X_{23} \vee \mathsf{AF}_{pqr} X_{234} \vee Y \\
X_{23} & \stackrel{\mu}{=} \mathsf{AF}_p X_4 \vee \mathsf{AF}_q X_3 \vee \mathsf{AF}_r X_2 \vee \mathsf{AF}_{pq} X_{34} \vee \mathsf{AF}_{pr} X_{24} \vee \mathsf{AF}_{qr} X_{23} \vee \mathsf{AF}_{pqr} X_{234} \vee Y \\
X_{234} & \stackrel{\mu}{=} \mathsf{AF}_p X_4 \vee \mathsf{AF}_q X_3 \vee \mathsf{AF}_r X_2 \vee \mathsf{AF}_{pq} X_{34} \vee \mathsf{AF}_{pr} X_{24} \vee \mathsf{AF}_{qr} X_{23} \vee \mathsf{AF}_{pqr} X_{234} \vee Y \\
X_{24} & \stackrel{\mu}{=} \mathsf{AF}_p X_4 \vee \mathsf{AF}_q X_3 \vee \mathsf{AF}_r X_2 \vee \mathsf{AF}_{pq} X_{34} \vee \mathsf{AF}_{pr} X_{24} \vee \mathsf{AF}_{qr} X_{23} \vee \mathsf{AF}_{pqr} X_{234} \vee Y \\
X_3 & \stackrel{\mu}{=} \mathsf{AF}_q X_3 \vee Y \\
X_{34} & \stackrel{\mu}{=} \mathsf{AF}_p X_4 \vee \mathsf{AF}_q X_3 \vee \mathsf{AF}_{pq} X_{34} \vee Y \\
X_4 & \stackrel{\mu}{=} \mathsf{AF}_p X_4 \vee Y
\end{array}\right\}$$

Figure 10: Determinized Mes produced from $\{X \stackrel{\mu}{=} \mathsf{AF}_{(q|p^*)^*.(qr^*).(p^*|q^*)} Y\}$

$$\left\{\begin{array}{ll}
X_1 & \stackrel{\mu}{=} \mathsf{AF}_p X_1 \vee \mathsf{AF}_q X_{12} \\
X_{12} & \stackrel{\mu}{=} \mathsf{AF}_p X_{14} \vee \mathsf{AF}_q X_{12} \vee \mathsf{AF}_r X_2 \vee \mathsf{AF}_{pr} X_{12} \vee Y \\
X_{14} & \stackrel{\mu}{=} \mathsf{AF}_p X_{14} \vee \mathsf{AF}_q X_{12} \vee Y \\
X_2 & \stackrel{\mu}{=} \mathsf{AF}_p X_4 \vee \mathsf{AF}_q X_3 \vee \mathsf{AF}_r X_2 \vee \mathsf{AF}_{pq} X_{34} \vee Y \\
X_3 & \stackrel{\mu}{=} \mathsf{AF}_q X_3 \vee Y \\
X_{34} & \stackrel{\mu}{=} \mathsf{AF}_p X_4 \vee \mathsf{AF}_q X_3 \vee \mathsf{AF}_{pq} X_{34} \vee Y \\
X_4 & \stackrel{\mu}{=} \mathsf{AF}_p X_4 \vee Y
\end{array}\right\} \qquad \left\{\begin{array}{ll}
X_1 & \stackrel{\mu}{=} \mathsf{AF}_p X_1 \vee \mathsf{AF}_q X_{12} \\
X_{12} & \stackrel{\mu}{=} \mathsf{AF}_r X_2 \vee \mathsf{AF}_{pr} X_{12} \vee X_{14} \\
X_{14} & \stackrel{\mu}{=} \mathsf{AF}_p X_{14} \vee \mathsf{AF}_q X_{12} \vee Y \\
X_2 & \stackrel{\mu}{=} \mathsf{AF}_r X_2 \vee X_{34} \\
X_{34} & \stackrel{\mu}{=} \mathsf{AF}_{pq} X_{34} \vee X_3 \vee X_4 \\
X_3 & \stackrel{\mu}{=} \mathsf{AF}_q X_3 \vee Y \\
X_4 & \stackrel{\mu}{=} \mathsf{AF}_p X_4 \vee Y
\end{array}\right\}$$

Figure 11: Determinized Mes of $\{X \stackrel{\mu}{=} \mathsf{AF}_{(q|p^*)^*.(qr^*).(p^*|q^*)} Y\}$ after simplifications

The determinization of a guarded potentiality Mes defined above is similar to the subset construction procedure used for determinizing finite automata [2]. In the worst-case, the size of the determinized Mes resulting from the translation of an equation block $\{X \stackrel{\mu}{=} \mathsf{AF}_\rho Y\}$ is exponential w.r.t. the size (number of operators and atomic propositions) of the regular formula ρ. However in practice, the size of determinized Mess obtained after simplifications is close to linear w.r.t. the size of ρ, as illustrated by the final Mes shown in Figure 11. When ρ is *deterministic* (*i.e.*, each atomic proposition occurs only once in the right-hand side of each equation of the guarded potentiality Mes used as intermediate form, and all atomic propositions are mutually exclusive on the states of the Kripke structure), the size of the resulting determinized Mes remains linear w.r.t. the size of ρ.

3.3.3. Operators $\mathsf{EF}^{\infty}_{\rho}$, $\mathsf{AF}^{\infty}_{\rho}$, $\mathsf{EG}^{\dashv}_{\rho}$, *and* $\mathsf{AG}^{\dashv}_{\rho}$

According to the rules given in Fig 4, the $\mathsf{EF}^{\infty}_{\rho}$ and $\mathsf{AF}^{\infty}_{\rho}$ operators are translated into equation blocks of the form $\{X \stackrel{\nu}{=} \mathsf{EF}_{\rho} X\}$ and $\{X \stackrel{\nu}{=} \mathsf{AF}_{\rho} X\}$, respectively. The interpretation of these equation blocks is given by $\nu\Phi_e$ and $\nu\Phi_a$, where the functionals $\Phi_e, \Phi_a : 2^S \to 2^S$ are defined as follows:

$$\Phi_e(U) = [\![\mathsf{EF}_{\rho} X]\!]_K[U/X] = ([\![\{X_1 \stackrel{\mu}{=} \mathsf{EF}_{\rho} X\}]\!]_K[U/X])(X_1)$$
$$\Phi_a(U) = [\![\mathsf{AF}_{\rho} X]\!]_K[U/X] = ([\![\{X_1 \stackrel{\mu}{=} \mathsf{AF}_{\rho} X\}]\!]_K[U/X])(X_1).$$

The evaluation of the $\mathsf{EF}^{\infty}_{\rho}$ and $\mathsf{AF}^{\infty}_{\rho}$ operators requires to compute the maximal fixed points of the functionals Φ_e and Φ_a, which are defined as the minimal fixed points of the functionals associated to the RESs $R_e = \{X_1 \stackrel{\mu}{=} \mathsf{EF}_{\rho} X\}$ and $R_a = \{X_1 \stackrel{\mu}{=} \mathsf{AF}_{\rho} X\}$. Therefore, these operators belong to $L\mu_2$, the μ-calculus fragment of alternation depth 2 [37], which allows one level of mutual recursion between minimal and maximal fixed points. The operators $\mathsf{EG}^{\dashv}_{\rho}$ and $\mathsf{AG}^{\dashv}_{\rho}$ are handled dually w.r.t. $\mathsf{AF}^{\infty}_{\rho}$ and $\mathsf{EF}^{\infty}_{\rho}$, respectively.

4. An on-the-fly model checker for CTRL

The translation from CTRL to MESs presented in Section 3 provides the basis of a model checking procedure, which was implemented by reusing as much as possible the on-the-fly verification technology available in the CADP toolbox [44]. The resulting CTRL model checker was coupled with the qualitative simulation tool GNA [12], which was enhanced in order to allow the specification of biological properties as temporal logic formulas. We outline below the principles of the model checker and its interconnection with GNA and CADP.

4.1. General architecture

The overall approach that we propose for analyzing the dynamic behaviour of GRNs is illustrated on Figure 12. Starting from the abstract descriptions of a GRN and of a biological property of interest, the modeler formally specifies, on one hand, the GRN behaviour as a system of piecewise-linear differential equations together with its initial conditions and, on the other hand, a temporal logic formula encoding the property. By performing a qualitative simulation of the network by means of the piecewise-linear model, GNA produces a Kripke structure representing an abstraction of the GRN behaviour, which is subsequently converted into an LTS. The temporal

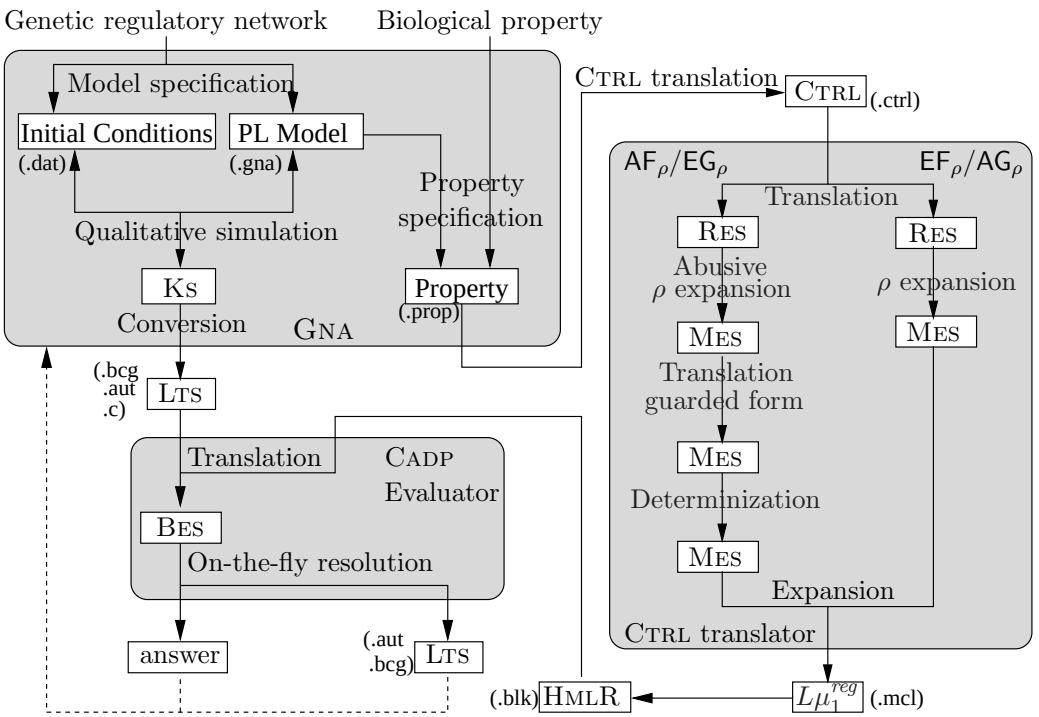

Figure 12: Ctrl translator and its interconnection to Gna and Cadp

logic property is also translated automatically by Gna, taking into account information present in the piecewise-linear model, into a Ctrl state formula. This formula is fed to the Ctrl translator, which produces a Mes and converts it into a HmlR specification. Finally, this specification and the Lts are given as input to the Evaluator 3.6 model checker of Cadp, which carries out the verification and produces a verdict accompanied by a diagnostic, *i.e.*, a subgraph of the Lts explaining the verification result. The modeler can then analyze the diagnostic using Gna in order to understand the truth value of the temporal property on the Kripke structure corresponding to the Grn and, if needed, to change the piecewise-linear model appropriately.

4.2. Qualitative simulation of genetic regulatory networks using GNA

The method we use for analyzing the dynamic behaviour of Grns relies upon a special class of *piecewise-linear differential equation* models, originally introduced by Glass and Kauffman [46]. The piecewise-linear models

provide a coarse-grained picture of the dynamics of GRNs, well-adapted to the current lack of quantitative information on many networks of interest. The models associate a protein concentration variable to each of the genes in the network, and capture the switch-like character of gene regulation by means of step functions that change their value at a threshold concentration of the proteins. The thresholds of the concentration variables define a hyperrectangular partition of the state space, such that in every region not located on a threshold hyperplane, the step functions evaluate to 0 or 1, and the piecewise-linear model reduces to an analytically solvable system of differential equations. More precisely, in such a region D we have

$$\frac{dx}{dt} = \kappa^D - \gamma^D\, x, \tag{1}$$

where $x \in \mathbb{R}^n_+$ is a vector of protein concentrations, $\kappa^D \in \mathbb{R}^n_+$ is a vector of (sums of) protein synthesis rate constants, and $\gamma^D \in \mathbb{R}^n_+ \times \mathbb{R}^n_+$ is a diagonal matrix of (sums of) degradation rate constants. It can be easily shown that in every D the solutions of (1) monotonically converge towards a focal point $(\gamma^D)^{-1}\,\kappa^D$. These convergence results can be generalized to the case of regions located on a threshold hyperplane [11, 34].

The qualitatively homogeneous behaviour inside regions D motivates the use of discrete abstractions, converting the continuous dynamics of the piecewise-linear models to discrete transition systems that are formally equivalent to Kripke structures (see [4] for a review and [3, 26, 55, 82] for some recent examples). We follow the approach developed in [11], which has been specifically defined for the class of piecewise-linear models considered here (see also [1, 45]). The states of the resulting Kripke structure correspond to regions in the state space, while the transitions arise from solutions of the piecewise-linear model that enter one region from another. The atomic propositions describe, among other things, the concentration bounds defining a region and the trend of the variables inside the region (increasing, decreasing, or steady).

Interestingly, it can be shown that the Kripke structure, and thus the qualitative dynamics of the system, are completely determined by inequality constraints imposing a total ordering on the threshold parameters and the focal parameters κ^D_i/γ^D_i for each variable x_i, $1 \leq i \leq n$. The definition of these constraints can generally be inferred from available data in the experimental literature or by intuitive reasoning, even in the absence of quantitative information on parameter values. The inequality constraints are used by

the computer tool GNA[1] (*Genetic Network Analyzer*) [12] to symbolically compute the Kripke structure for a given piecewise-linear model. We call the process of computing a Kripke structure containing the states reachable from given initial conditions a qualitative simulation of the network. GNA has been used for the qualitative simulation of a number of bacterial regulatory networks, such as the initiation of sporulation in *Bacillus subtilis* [33], quorum sensing in *Pseudomonas aeruginosa* [83], the carbon starvation response in *Escherichia coli* [73], and the onset of virulence in *Erwinia chrysanthemi* [75].

In order to analyze the Kripke structures generated by qualitative simulation, GNA has been enhanced to support formal verification of biological properties by means of CTRL, resulting in the new version 7.0. First of all, the Kripke structure can be exported to the LTS format of CADP by applying the conversion given in [28], which consists in moving the atomic propositions valid at a state s on the actions labeling the transitions going out of s. Second, the user can specify biological properties of interest using a dedicated property editor, which provides pattern-based property specification support, following the query schemes proposed in [71], based on a review of frequently-asked questions by modelers. The translation of the patterns into CTRL is shown in Table 1. An alternative to the pattern-based property editor, would be to use live sequence charts [31], providing a graphical property specification tool [61]. Expert users can also directly specify CTRL formulas of arbitrary complexity that are not covered by the patterns. The properties can be exported to a file that the CTRL translator is able to translate into the HMLR format accepted by CADP. In order to achieve a tighter integration between the two tools, and allow the modeler to analyze the verification results in the GNA environment, a Web server-based connection between the tools has been implemented (Monteiro *et al.*, in preparation).

4.3. On-the-fly CTRL model checking using CADP

The most direct way of obtaining a model checker for CTRL was to take advantage of existing verification technology. As verification engine, we use CADP[2] (*Construction and Analysis of Distributed Processes*) [44], a state-of-the-art verification toolbox for concurrent asynchronous systems. CADP

[1] GNA is distributed by the company Genostar and freely available for non-profit academic research purposes at `http://ibis.inrialpes.fr/gna`.

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

Pattern	CTRL formula
Occurrence/Exclusion pattern	
It **is possible** for a state φ to occur	$\mathsf{EF}_{\mathsf{true}^*}\varphi$
It **is not possible** for a state φ to occur	$\neg\mathsf{EF}_{\mathsf{true}^*}\varphi$
Consequence pattern	
If a state φ occurs, then it is **possibly** followed by a state ψ	$\mathsf{AG}_{\mathsf{true}^*}(\varphi \Rightarrow \mathsf{EF}_{\mathsf{true}^*}\psi)$
If a state φ occurs, then it is **necessarily** followed by a state ψ	$\mathsf{AG}_{\mathsf{true}^*}(\varphi \Rightarrow \mathsf{AF}_{\mathsf{true}^*}\psi)$
Sequence pattern	
A state ψ is reachable and is **possibly** preceded **at some time** by a state φ	$\mathsf{EF}_{\mathsf{true}^*.\varphi.\mathsf{true}^*}\psi$
A state ψ is reachable and is **possibly** preceded **all the time** by a state φ	$\mathsf{EF}_{\varphi^*}\psi$
A state ψ is reachable and is **necessarily** preceded **at some time** by a state φ	$\mathsf{EF}_{\mathsf{true}^*}\psi \wedge \neg\mathsf{EF}_{(\neg\varphi)^*}\psi$
A state ψ is reachable and is **necessarily** preceded **all the time** by a state φ	$\mathsf{EF}_{\mathsf{true}^*}\psi \wedge \mathsf{AG}_{\mathsf{true}^*}(\neg\varphi \Rightarrow \mathsf{AG}_{\mathsf{true}^*}\neg\psi)$
Invariance pattern	
A state φ **can** persist indefinitely	$\mathsf{EG}_{\mathsf{true}^*}\varphi$
A state φ **must** persist indefinitely	$\mathsf{AG}_{\mathsf{true}^*}\varphi$

Table 1: Rules for translating the patterns (with all their variations) into CTRL

offers a wide range of functionalities assisting the user throughout the design process: compilation and rapid prototyping, random execution, interactive and guided simulation, model checking and equivalence checking, test generation, and performance evaluation. The toolbox accepts as input process algebraic descriptions in LOTOS [54] or CHP [66], as well as networks of communicating automata in the EXP language [62].

The tools of CADP 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 [43]. CADP contains the on-the-fly model checker EVALUATOR 3.6 [69], 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 boolean equation system, which is performed using the algorithms available in the generic CÆSAR_SOLVE library [67]. EVALUATOR 3.6 uses HMLR as intermediate language: $L\mu_1^{reg}$ formulas are translated into HMLR specifications, whose evaluation on implicit LTSs can be straightforwardly encoded as a local boolean equation system resolution [29, 69]. The tool also generates full diagnostics (examples and counterexamples) illustrating the truth value of the formulas.

In order to reuse 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 3.6. This technical choice motivated the definition of the translation from CTRL to

MESs in the first place. The architecture of the CTRL translator (about 12,000 lines of code) is shown in Figure 12. The tool takes as input a CTRL state formula and translates it to a MES following the phases described in Section 3, which are different for the EF_ρ and AF_ρ operators and their dual counterparts. The MES obtained is then converted into a HMLR specification by expanding the basic CTRL temporal operators in terms of HML modalities as described in [68].

Complexity. Table 2 summarizes the complexity of our model checking procedure for CTRL. The EF_ρ and EF^∞_ρ operators, together with their respective duals AG_ρ and $\mathsf{AG}^\dashv_\rho$, are evaluated in linear-time w.r.t. the size of the formula and the size of the Kripke structure by applying the boolean equation system resolution algorithms given in [67, 70]. Moreover, the evaluation of these operators has a memory complexity $O(|\rho| \cdot |S|)$, meaning that only the states (and not the transitions) of the Kripke structure are stored. This fragment of CTRL is the state-based counterpart of PDL-Δ [77], which is more expressive than CTL* [36]. Of course, this does not yield a linear-time model checking procedure for CTL* (nor for its fragment LTL), because the translation from CTL* to PDL-Δ is not succinct [85]. However, the linear-time evaluation of the EF^∞_ρ operator allows an efficient detection of complex cycles describing oscillation properties [22].

Table 2: Complexity of model checking CTRL operators on $K = \langle S, P, L, T, s_0 \rangle$

Operator		Complexity	
		ρ deterministic	ρ nondeterministic
EF_ρ	AG_ρ	$O(\|\rho\| \cdot (\|S\| + \|T\|))$	
AF_ρ	EG_ρ	$O(\|\rho\| \cdot (\|S\| + \|T\|))$	$O(2^{\|\rho\|} \cdot (\|S\| + \|T\|))$
EF^∞_ρ	$\mathsf{AG}^\dashv_\rho$	$O(\|\rho\| \cdot (\|S\| + \|T\|))$	
AF^∞_ρ	$\mathsf{EG}^\dashv_\rho$	$O(\|\rho\| \cdot (\|S\| + \|T\|))$	$O(2^{2\|\rho\|} \cdot (\|S\| + \|T\|)^2)$

The AF_ρ operator and its dual EG_ρ are evaluated in linear-time only when the regular subformula ρ is deterministic. In the general case, these operators are evaluated in exponential-time w.r.t. the size of ρ (because of the determinization phase) but still in linear-time in the size of the Kripke structure. In practice, the size of temporal formulas is much smaller than the size of Kripke structures, which reduces the impact of the factor $2^{|\rho|}$ on the

total cost of model checking. Finally, the $\mathsf{AF}^{\infty}_{\rho}$ operator and its dual $\mathsf{EG}^{\dashv}_{\rho}$ are evaluated in linear-time when ρ is deterministic by applying a symmetric version of the boolean equation system resolution algorithm in [70]; 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 size of the Kripke structure. This complexity seems difficult to lower, since the boolean equation systems produced by translating these operators are of alternation depth 2 and have a general shape (arbitrary amounts of disjunctive and conjunctive equations).

5. Verification of genetic regulatory networks

5.1. Model of carbon starvation response in E. coli

To assess the applicability of CTRL, we have analyzed a model of the carbon starvation response in the bacterium *E. coli*. In the absence of essential carbon sources in its growth environment, an *E. coli* population abandons exponential growth and enters a non-growth state called stationary phase. This growth-phase transition is accompanied by numerous physiological changes in the bacteria [52], and controlled by a complex genetic regulatory network integrating various environmental signals. A key part of this network is shown in Figure 13.

The molecular basis of the adaptation of the growth of *E. coli* to the nutritional conditions has been the focus of extensive studies for decades [48, 50]. However, notwithstanding the enormous amount of information accumulated on the genes, proteins, and other molecules known to be involved in the stress adaptation process, it is currently not understood how the response of the cell emerges from the regulatory network. Moreover, with some exceptions [17, 24], numerical values for the kinetic parameters and the molecular concentrations are absent, which makes it difficult to apply traditional methods for the dynamical modeling of GRNs.

This has motivated the development of a piecewise-linear model of the carbon starvation network, using the tools presented in Section 4. More precisely, the dynamics of the network are described by 9 coupled piecewise-linear differential equations, and 59 inequality constraints on the parameter values. The qualitative analysis of a network of this size and complexity generates huge Kripke structures: the entire state set consists of approximately $\mathcal{O}(10^{10})$ states, while the subset of states that is most relevant for our purpose, *i.e.*, the states that are reachable from an initial state corresponding to a particular growth state of the bacteria, still consists of $\mathcal{O}(10^{4})$ states.

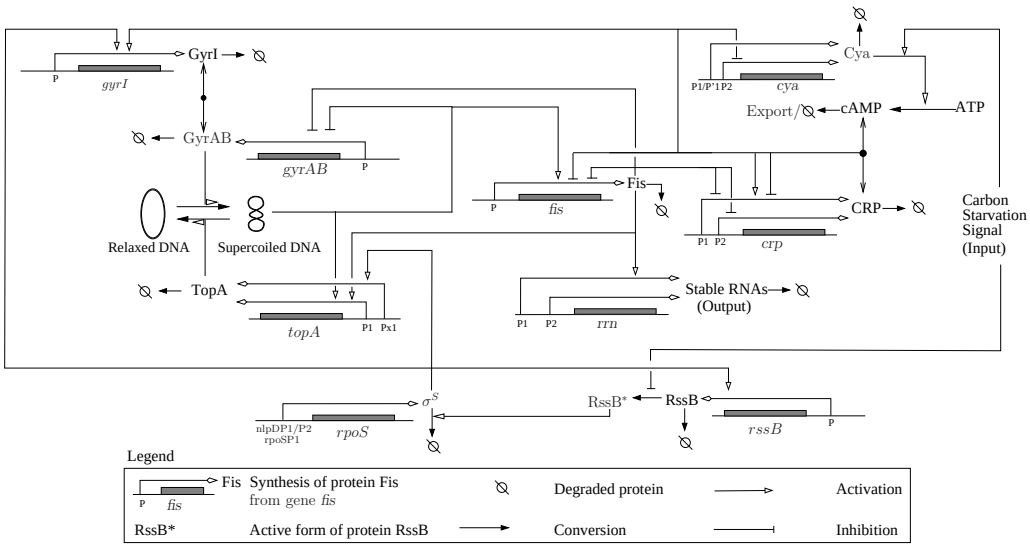

Figure 13: Network of key genes, proteins, and regulatory interactions involved in the GRN controlling the carbon starvation response in *E. coli* [73].

It is obvious that Kripke structures of this size cannot be analyzed by visual inspection, and that formal verification techniques are needed to get a better insight into the transient and asymptotic dynamics of the network.

5.2. Analysis of carbon starvation response model

A first question of interest concerns the attractors of the system and their reachability. The first attractor identified by means of GNA is a state in the Kripke structure corresponding to an asymptotically stable steady state of the piecewise-linear model. This state is characterized by a low basal expression level of the stable RNAs, an indicator of growth arrest and thus of stationary phase. The second attractor corresponds to another steady state of the piecewise-linear model, which is also stable as shown by the mathematical analysis in [47]. It has a high expression level of the stable RNAs, characteristic of the high growth rate in exponential phase. The steady state is asymptotically reached by means of damped oscillations of the concentrations of some of the proteins and the stable RNAs. Under the discrete abstraction of the piecewise-linear dynamics, the damped oscillations appear as a terminal cycle in the Kripke structure (the eventual transition to the

stable steady state is ignored as it does not occur in a finite number of transitions).

We label the above attractors a_{stat} (the stationary-phase steady state) and a_{exp} (the damped oscillations leading to the exponential-phase steady state) and check the following property:

$$\mathsf{AG}_{\mathsf{true}^*}(\mathsf{EF}_{\mathsf{true}^*}\, a_{exp} \vee \mathsf{EF}_{\mathsf{true}^*}\, a_{stat}) \tag{2}$$

The CTRL model checker returns true, confirming that a_{stat} and a_{exp} are the only attractors of the system. Can they both occur for a given input, that is, under fixed nutrient conditions? This would show that the system is bistable. We introduce the atomic proposition *sig* to denote the carbon starvation input signal, and specify the following CTRL property:

$$(\mathsf{EF}_{sig^*}\, a_{exp} \wedge \mathsf{EF}_{sig^*}\, a_{stat}) \vee (\mathsf{EF}_{\neg sig^*}\, a_{exp} \wedge \mathsf{EF}_{\neg sig^*}\, a_{stat}) \tag{3}$$

Both are false, indicating that for given nutrient conditions the attractors are mutually exclusive. More precisely, if *sig* is present (nutrient depletion) then the stationary-phase attractor is inevitably reached, whereas in the absence of *sig* (nutrient availability), the system necessarily evolves towards the exponential-phase attractor. That is, the model checker returns true for the following property:

$$\mathsf{AG}_{\mathsf{true}^*}((sig \Rightarrow \mathsf{AF}_{sig^*}\, a_{stat}) \wedge (\neg sig \Rightarrow \mathsf{AF}_{\neg sig^*}\, a_{exp})) \tag{4}$$

In mathematical terms, the system is therefore not bistable, but has a monostable steady-state response to each of the two possible inputs (nutrient depletion *vs.* nutrient availability).

The damped oscillations predicted by the model are an unexpected and hitherto unobserved phenomenon. It would therefore be interesting to better characterize the predictions on the molecular level. We have computed by means of GNA the part of the Kripke structure reachable from an initial state corresponding to the stationary-phase steady state with the stress signal switched off ($\neg sig$). This initial state mimicks the physiological state of the bacteria at the nutrient upshift. The resulting Kripke structure consists of 9603 states.

We first check which concentrations are oscillating in the terminal cycle a_{exp}, for each of the proteins and the stable RNAs. Let *dec_rrn* (*inc_rrn*) represent a decreasing (increasing) concentration of stable RNAs (which are

transcribed from the *rrn* operons). The following CTRL property is true, confirming the predicted (damped) oscillations for the concentration of stable RNAs:

$$\mathsf{EF}_{\mathsf{true}^*}\mathsf{EF}^{\infty}_{a^+_{exp}.(inc_rrn^+.\mathsf{true}^*.dec_rrn^+)^+} \tag{5}$$

A similar property is true for GyrAB and Fis, regulators of both the DNA supercoiling level and the accumulation of stable RNAs in the cell. It points at the role of the negative feedback loop involved in the homeostatic control of the DNA supercoiling level in causing the damped oscillations [73].

We can check a stricter property by requiring that all paths in the Kripke structure lead to the terminal cycle with an oscillating stable RNA concentration:

$$\mathsf{AG}_{\mathsf{true}^*}\mathsf{EF}^{\infty}_{a^+_{exp}.(inc_rrn^+.\mathsf{true}^*.dec_rrn^+)^+} \tag{6}$$

Property (7) is false, confirming that some paths do not end up in this cycle. This can be explained by the fact that the Kripke structure contains several nonterminal cycles in addition to the terminal cycle a_{exp}. However, we can impose a path restriction on the AG operator to guide the model checker towards the terminal cycle:

$$\mathsf{AG}_{true^*.a_{exp}}\mathsf{EF}^{\infty}_{a^+_{exp}.(inc_rrn^+.\mathsf{true}^*.dec_rrn^+)^+} \tag{7}$$

The CTRL model checker returns true for property (8), proving that all paths satisfying the restriction reach the terminal cycle where the concentration of stable RNAs continues to oscillate. Interestingly, the following property is also true:

$$\mathsf{AG}_{true^*.dec_Crp^+.a_{exp}}\mathsf{EF}^{\infty}_{a^+_{exp}.(inc_rrn^+.\mathsf{true}^*.dec_rrn^+)^+} \tag{8}$$

This suggests that the decrease of the concentration of Crp drives the system towards the terminal cycle.

The CTRL formulas above illustrate the usefulness of regular expressions for characterizing sequences of states in a concise manner. The nested iteration operators present in the regular formulas make the CTRL formulas unexpressible using standard temporal logics such as CTL or LTL. In addition, the $\mathsf{EF}^{\infty}_{\rho}$ operator enables a natural formulation of infinite repetitions of sequences defined by ρ, such as those corresponding to (damped) oscillations in the *E. coli* example.

6. Conclusions and future work

Applications of model checking in systems biology have demonstrated its usefulness for understanding the dynamic behaviour of regulatory networks in living cells, but also outlined 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. The temporal logic CTRL that we proposed, an extension of CTL with regular expressions and fairness operators, allows a natural and concise description of typical properties of biological interest, such as the presence of multistability or oscillations in the concentrations of molecular species. We were able 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 CADP. This modular architecture allowed us to reduce the development effort and to take advantage of existing, robust model checking technology.

The extension of classical temporal logics with regular language constructs to increase their expressiveness and user-friendliness is a long-standing line of research. One of the first proposals in this direction was ETL [86], an extension of LTL with regular grammars, which is strictly more expressive than LTL while still having the same complexity of evaluation on Kripke structures. Another manner of increasing expressiveness is to enhance temporal operators with automata on infinite sequences; this was attempted for CTL [49] and CTL* [81]. Despite their expressive power, these extensions are difficult to implement and use in practice because of their complex syntax.

A more user-friendly approach, which led to successful implementations, is to index temporal operators by regular expressions instead of automata. FORSPEC [6] and EAGLE [9] are extensions of LTL with regular expressions and data handling mechanisms, dedicated respectively to hardware and runtime verification. RCTL [15] is an extension of CTL with regular expressions, which served subsequently as basis for the SUGAR [14] and PSL [53] specification languages used for hardware verification. RegCTL [19] is another extension of CTL with regular expressions, more expressive than RCTL, obtained by indexing the Until operator of CTL with regular expressions. Our proposal is in line with these latter approaches, but focuses on the translation of CTRL to the modal μ-calculus, which among other things allows us to reuse the on-the-fly verification technology available for the latter formalism.

This contrasts with the model checking approaches proposed for the other extensions of CTL, which are most of the time based on automata.

In this paper, we have employed CTRL for the verification of dynamic properties of GRNs modeled by piecewise-linear differential equations. The continuous dynamics of these models can be converted into discrete state transition graphs that are formally equivalent to Kripke structures. The computer tool GNA is able to generate the state transition graphs and export them as Kripke structures to CADP. This allows the use of CADP for verifying CTRL properties on GRNs, as illustrated on the carbon starvation network in *E. coli*.

The application of CTRL in systems biology and bioinformatics is not restricted to the class of models considered in this paper though. CTRL is interpreted on Kripke structures, which provide a general description of dynamical systems that implicitly or explicitly underlie many of the existing discrete formalisms used for the modeling of regulatory networks in the cell, such as Boolean networks and their generalizations, Petri nets, and process algebras [23, 40]. In addition, other types of continuous models of regulatory networks, by defining appropriate discrete abstractions, can possibly be mapped to Kripke structures as well. As a consequence, CTRL can be combined with many of the other approaches proposed for the application of formal verification tools to biological regulatory networks [5, 8, 12, 16, 21, 22, 41].

We plan to continue our work on several directions. First, we will extend the CÆSAR_SOLVE [67] library of CADP with resolution algorithms handling boolean equation systems of alternation depth 2 [84] in order to obtain an on-the-fly evaluation of the $\mathsf{AF}^{\infty}_{\rho}$ 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 boolean equation system resolution algorithms proposed in [56, 57]. Fourth, we will develop tools to further support non-expert users in applying formal verification to the analysis of biological regulatory networks ([71], Monteiro *et al.*, in preparation).

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). Estelle

Dumas is grateful to David Champelovier, Hubert Garavel, Romain Lacroix, and Michel Page for their valuable assistance in developing the translator from CTRL to HMLR.

References

[1] Adélade, M., Sutre, G., 2004. Parametric analysis and abstraction of genetic regulatory networks. In: Proc. BioCONCUR 04. London, UK.

[2] Aho, A. V., Sethi, R., Ullman, J. D., 1986. Compilers: Principles, Techniques and Tools. Addison-Wesley.

[3] Alur, R., Dang, T., Ivančić, F., 2006. Counter-example guided predicate abstraction of hybrid systems. Theor. Comput. Sci. 354 (2), 250–271.

[4] Alur, R., Henzinger, T., Lafferriere, G., Pappas, G., 2000. Discrete abstractions of hybrid systems. Proceedings of the IEEE 88 (7), 971–984.

[5] Antoniotti, M., Policriti, A., Ugel, N., Mishra, B., 2003. Model building and model checking for biochemical processes. Cell Biochemistry and Biophysics 38 (3), 271–286.

[6] Armoni, R., Fix, L., Flaisher, A., Gerth, R., Ginsburg, B., Kanza, T., Landver, A., Mador-Haim, S., Singerman, E., Tiemeyer, A., Vardi, M. Y., Zbar, Y., April 2002. The ForSpec temporal logic: A new temporal property-specification language. In: Katoen, J.-P., Stevens, P. (Eds.), Proceedings of the 8th International Conference on Tools and Algorithms for the Construction and Analysis of Systems TACAS'02 (Grenoble, France). Vol. 2280 of Lecture Notes in Computer Science. Springer Verlag, pp. 296–211.

[7] Ballarini, P., Mazza, T., Palmisano, A., Csikasz-Nagy, A., 2009. Studying irreversible transitions in a model of cell cycle regulation. In: Proceedings of the third International Workshop on Practical Applications of Stochastic Modelling, PASM 2008.

[8] Barnat, J., Brim, L., Cerná, I., Drazan, S., Safranek, D., 2008. Parallel model checking large-scale genetic regulatory networks with DiVinE. In: From Biology to Concurrency and Back, FBTC 2007. Vol. 194 of Electronic Notes in Theoretical Computer Science.

[9] Barringer, H., Goldberg, A., Havelund, K., Sen, K., January 2004. Rule-based runtime verification. In: Steffen, B., Levi, G. (Eds.), Proceedings of the 5th International Conference on Verification, Model Checking, and Abstract Interpretation VMCAI'04 (Venice, Italy). Vol. 2937 of Lecture Notes in Computer Science. Springer Verlag, pp. 44–57.

[10] Batt, G., Bergamini, D., de Jong, H., Gavarel, H., Mateescu, R., 2004. Model checking genetic regulatory networks using GNA and CADP. In: Graf, S., Mounier, L. (Eds.), Eleventh International SPIN Workshop on Model Checking of Software, SPIN 2004. Vol. 2989 of Lecture Notes in Computer Science. Springer-Verlag, Berlin, pp. 158–163.

[11] Batt, G., de Jong, H., Page, M., Geiselmann, J., 2008. Symbolic reachability analysis of genetic regulatory networks using discrete abstractions. Automatica 44 (4), 982–989.

[12] Batt, G., Ropers, D., de Jong, H., Geiselmann, J., Mateescu, R., Page, M., Schneider, D., 2005. 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.

[13] Batt, G., Yordanov, B., Belta, C., Weiss, R., 2007. Robustness analysis and tuning of synthetic gene networks. Bioinformatics 23 (18), 2415–2422.

[14] Beer, I., Ben-David, S., Eisner, C., Fisman, D., Gringauze, A., Rodeh, Y., July 2001. The temporal logic Sugar. In: Berry, G., Comon, H., Finkel, A. (Eds.), Proceedings of the 13th International Conference on Computer Aided Verification CAV'2001 (Paris, France). Vol. 2102 of Lecture Notes in Computer Science. Springer Verlag, pp. 363–367.

[15] Beer, I., Ben-David, S., Landver, A., June 1998. On-the-fly model checking of RCTL formulas. In: Hu, A., Vardi, M. Y. (Eds.), Proceedings of the 10th International Conference on Computer Aided Verification CAV'98 (Vancouver, BC, Canada). Vol. 1427 of Lecture Notes in Computer Science. Springer Verlag, pp. 184–194.

[16] Bernot, G., Comet, J.-P., Richard, A., Guespin, J., 2004. Application of formal methods to biological regulatory networks: Extending Thomas'

asynchronous logical approach with temporal logic. Journal of Theoretical Biology 229 (3), 339–348.

[17] Bettenbrock, K., Fischer, S., Kremling, A., Jahreis, K., Sauter, T., Gilles, E.-D., 2006. A quantitative approach to catabolite repression in *Escherichia coli*. J. Biol. Chem. 281 (5), 2578–84.

[18] Bosnacki, D., ten Eikelder, H., Steijaert, M., de Vink, E., 2008. Stochastic analysis of amino acid substitution in protein synthesis. In: Heiner, M., Uhrmacher, A. (Eds.), Computational Methods in Systems Biology, CMSB-08. Vol. 5307 of Lecture Notes in Computer Science. Springer-Verlag, Berlin, pp. 367–386.

[19] Brázdil, T., Cerná, I., 2006. Model checking of RegCTL. Computers and Artificial Intelligence 25 (1).

[20] Brzozowski, J. A., 1964. Derivatives of regular expressions. J. ACM 11 (4), 481–494.

[21] Calder, M., Vyshemirsky, V., Gilbert, D., Orton, R., 2005. Analysis of signalling pathways using the PRISM model checker. In: Plotkin, G. (Ed.), Computational Methods in Systems Biology, CMSB-05. Edinburgh, Scotland, pp. 79–90.

[22] Chabrier-Rivier, N., Chiaverini, M., Danos, V., Fages, F., Schächter, V., 2004. Modeling and querying biomolecular interaction networks. Theoretical Computer Science 325 (1), 25–44.

[23] Chaouiya, C., 2007. Petri net modelling of biological networks. Briefings in Bioinformatics 8 (4), 210–219.

[24] Chassagnole, C., Noisommit-Rizzi, N., Schmid, J., Mauch, K., Reuss, M., 2002. Dynamic modeling of the central carbon metabolism of *Escherichia coli*. Biotechnology and Bioengineering 79 (1), 53–73.

[25] Chen, K., Calzone, L., Csikasz-Nagy, A., Cross, F., Novak, B., Tyson, J., 2004. Integrative analysis of cell cycle control in budding yeast. Molecular Biology of the Cell 15 (8), 3841–3862.

[26] Chutinan, A., Krogh, B., 2001. Verification of infinite-state dynamic systems using approximate quotient transition systems. IEEE Trans. Autom. Control 46 (9), 1401–1410.

[27] Cimatti, A., Clarke, E., Giunchiglia, F., Roveri, M., Apr. 2000. NuSMV: a new symbolic model checker. Springer International Journal on Software Tools for Technology Transfer (STTT) 2 (4), 410–425.

[28] Clarke, E., Grumberg, O., Peled, D., 2000. Model Checking. MIT Press.

[29] Cleaveland, R., Steffen, B., April 1993. A linear-time model-checking algorithm for the alternation-free modal mu-calculus. Formal Methods in System Design 2 (2), 121–147.

[30] Dam, M., April 1994. CTL* and ECTL* as fragments of the modal μ-calculus. Theoretical Computer Science 126 (1), 77–96.

[31] Damm, W., Harel, D., 2001. Lscs: Breathing life into message sequence charts. Formal Methods in System Design 19 (1), 45–80.

[32] de Jong, H., 2002. Modeling and simulation of genetic regulatory systems: A literature review. Journal of Computational Biology 9 (1), 67–103.

[33] de Jong, H., Geiselmann, J., Batt, G., Hernandez, C., Page, M., 2004. Qualitative simulation of the initiation of sporulation in *B. subtilis*. Bull. Math. Biol. 66 (2), 261–299.

[34] de Jong, H., Gouzé, J.-L., Hernandez, C., Page, M., Sari, T., Geiselmann, J., 2004. Qualitative simulation of genetic regulatory networks using piecewise-linear models. Bulletin of Mathematical Biology 66 (2), 301–340.

[35] Dubnau, D., Losick, R., 2006. Bistability in bacteria. Molecular Microbiology 61 (3), 564–572.

[36] Emerson, E. A., Halpern, J. Y., January 1983. "Sometimes" and "not never" revisited: On branching versus linear time. In: Proceedings of the 10th Annual ACM Symposium on Principles of Programming Languages POPL'83 (Austin, Texas). pp. 127–140, also appeared in Journal of ACM, 33(1):151-178, 1986.

[37] Emerson, E. A., Lei, C.-L., 1986. Efficient model checking in fragments of the propositional mu-calculus. In: Proceedings of the 1st International Symposium on Logic in Computer Science LICS'86. pp. 267–278.

[38] Emerson, E. A., Lei, C.-L., 1987. Modalities for model checking: Branching time logic strikes back. Science of Computer Programming 8.

[39] Fischer, M. J., Ladner, R. E., September 1979. Propositional dynamic logic of regular programs. Journal of Computer and System Sciences 18 (2), 194–211.

[40] Fisher, J., Henzinger, T., 2007. Executable cell biology. Nature Biotechnology 25 (11), 1239–1250.

[41] Fisher, J., Piterman, N., Hajnal, A., Henzinger, T., 2007. Predictive modeling of signaling crosstalk during *C. elegans* vulval development. PLoS Computational Biology 3 (5), e92.

[42] Fisher, J., Piterman, N., Hubbard, E. J. A., Stern, M. J., Harel, D., 2005. Computational insights into *Caenorhabditis elegans* vulval development. Proceedings of the National Academy of Sciences of the USA 102 (6), 1951–1956.

[43] Garavel, H., March 1998. OPEN/CÆSAR: An open software architecture for verification, simulation, and testing. In: Steffen, B. (Ed.), Proceedings of the First International Conference on Tools and Algorithms for the Construction and Analysis of Systems TACAS'98 (Lisbon, Portugal). Vol. 1384 of Lecture Notes in Computer Science. Springer Verlag, Berlin, pp. 68–84, full version available as INRIA Research Report RR-3352.

[44] Garavel, H., Lang, F., Mateescu, R., Serwe, W., July 2007. CADP 2006: A toolbox for the construction and analysis of distributed processes. In: Damm, W., Hermanns, H. (Eds.), Proceedings of the 19th International Conference on Computer Aided Verification CAV'2007 (Berlin, Germany). Vol. 4590 of Lecture Notes in Computer Science. Springer Verlag, pp. 158–163.

[45] Ghosh, R., Tomlin, C., 2004. Symbolic reachable set computation of piecewise affine hybrid automata and its application to biological modelling: Delta-Notch protein signalling. IET Systems Biology 1 (1), 170–183.

[46] Glass, L., Kauffman, S., 1973. The logical analysis of continuous non-linear biochemical control networks. Journal of Theoretical Biology 39 (1), 103–129.

[47] Grognard, F., de Jong, H., Gouzé, J.-L., 2007. Piecewise-linear models of genetic regulatory networks: Theory and example. In: Queinnec, I., Tarbouriech, S., Garcia, G., Niculescu, S. (Eds.), Biology and Control Theory: Current Challenges. Vol. 357 of Lecture Notes in Control and Information Science. Springer-Verlag, Berlin, pp. 137–159.

[48] Gutierrez-Ríos, R., Freyre-Gonzalez, J., Resendis, O., Collado-Vides, J., Saier, M., Gosset, G., 2007. Identification of regulatory network topological units coordinating the genome-wide transcriptional response to glucose in *Escherichia coli*. BMC Microbiol. 7 (1), 53.

[49] Hamaguchi, K., Hiraishi, H., Yajima, S., June 1990. Branching time regular temporal logic for model checking with linear time complexity. In: Clarke, E. M., Kurshan, R. P. (Eds.), Proceedings of the 2nd International Conference on Computer Aided Verification CAV'90 (New Brunswick, New Jersey, USA). Vol. 531 of Lecture Notes in Computer Science. Springer Verlag, Berlin, pp. 253–262.

[50] Hengge-Aronis, R., 2000. The general stress response in *Escherichia coli*. In: Storz, G., Hengge-Aronis, R. (Eds.), Bacterial Stress Responses. ASM Press, Washington D.C., pp. 161–77.

[51] Holzmann, G., 2003. The SPIN Model Checker – Primer and Reference Manual. Addison-Wesley.

[52] Huisman, G., Siegele, D., Zambrano, M., Kolter, R., 1996. Morphological and physiological changes during stationary phase. In: Neidhardt, F. (Ed.), *Escherichia coli and Salmonella: Cellular and Molecular Biology*. ASM Press, Washington DC, pp. 1672–82.

[53] IEEE, September 2004. PSL: Property specification language. Standard P1850, IEEE Computer Society.

[54] ISO/IEC, September 1989. LOTOS — a formal description technique based on the temporal ordering of observational behaviour. International Standard 8807, International Organization for Standardization — Information Processing Systems — Open Systems Interconnection, Genève.

[55] Jha, S., Krogh, B., Weimer, J., Clarke, E., 2007. Reachability for linear hybrid automata using iterative relaxation abstraction. In: Bemporad, A., Bicchi, A., Buttazzo, G. (Eds.), Proc. HSCC 2007. Vol. 4416 of LNCS. Springer-Verlag, pp. 287–300.

[56] Joubert, C., Mateescu, R., February 2005. Distributed local resolution of boolean equation systems. In: Tirado, F., Prieto, M. (Eds.), Proceedings of the 13th Euromicro Conference on Parallel, Distributed and Network-Based Processing PDP'2005 (Lugano, Switzerland). IEEE Computer Society, pp. 264–271.

[57] Joubert, C., Mateescu, R., March–April 2006. Distributed on-the-fly model checking and test case generation. In: Valmari, A. (Ed.), Proceedings of the 13th International SPIN Workshop on Model Checking of Software SPIN'2006 (Vienna, Austria). Vol. 3925 of Lecture Notes in Computer Science. Springer Verlag, pp. 126–145.

[58] Kleene, S. C., 1952. Introduction to Metamathematics. North-Holland.

[59] Klipp, E., Nordlander, B., Krüger, R., Gennemark, P., Hohmann, S., 2005. Integrative model of the response of yeast to osmotic shock. Nature Biotechnology 23 (8), 975–982.

[60] Kozen, D., 1983. Results on the propositional μ-calculus. Theoretical Comput. Sci. 27, 333–354.

[61] Kugler, H., Harel, D., Pnueli, A., Lu, Y., Bontemps, Y., 2005. Temporal logic for scenario-based specifications. In: Halbwachs, N., Zuck, L. (Eds.), Proceedings of the 11th International Conference on Tools and Algorithms for the Construction and Analysis of Systems TACAS'05 (Edinburgh, U.K.). Vol. 3440 of Lecture Notes in Computer Science. Springer-Verlag, pp. 445–460.

[62] Lang, F., November 2005. EXP.OPEN 2.0: A flexible tool integrating partial order, compositional, and on-the-fly verification methods. In: van de Pol, J., Romijn, J., Smith, G. (Eds.), Proceedings of the 5th International Conference on Integrated Formal Methods IFM'2005 (Eindhoven, The Netherlands). Vol. 3771 of Lecture Notes in Computer Science. Springer Verlag, full version available as INRIA Research Report RR-5673.

[63] Larsen, K. G., March 1988. Proof systems for Hennessy-Milner logic with recursion. In: Proceedings of the 13th Colloquium on Trees in Algebra and Programming CAAP'88 (Nancy, France). Vol. 299 of Lecture Notes in Computer Science. Springer Verlag, Berlin, pp. 215–230.

[64] Leloup, J.-C., Goldbeter, A., 2003. Toward a detailed computational model for the mammalian circadian clock. Proceedings of the National Academy of Sciences of the USA 100 (12), 7051–7056.

[65] Manna, Z., Pnueli, A., 1992. The Temporal Logic of Reactive and Concurrent Systems, volume I: Specification. Springer Verlag.

[66] Martin, A. J., 1986. Compiling communicating processes into delay-insensitive VLSI circuits. Distributed Computing 1 (4), 226–234.

[67] Mateescu, R., February 2006. CÆSAR_SOLVE: A generic library for on-the-fly resolution of alternation-free boolean equation systems. Springer International Journal on Software Tools for Technology Transfer (STTT) 8 (1), 37–56, full version available as INRIA Research Report RR-5948, July 2006.

[68] Mateescu, R., Monteiro, P., Dumas, E., de Jong, H., Oct. 2008. Computation tree regular logic for genetic regulatory networks. In: Cha, S. D., Choi, J.-Y., Kim, M., Lee, I., Viswanathan, M. (Eds.), Proceedings of the 6th International Symposium on Automated Technology for Verification and Analysis ATVA'08 (Seoul, South Korea). Vol. 5311 of Lecture Notes in Computer Science. Springer Verlag, pp. 48–63, full version available as INRIA Research Report RR-6521.

[69] Mateescu, R., Sighireanu, M., March 2003. Efficient on-the-fly model-checking for regular alternation-free mu-calculus. Sci. Comput. Programming 46 (3), 255–281.

[70] Mateescu, R., Thivolle, D., May 2008. A model checking language for concurrent value-passing systems. In: Proceedings of the 15th International Symposium on Formal Methods FM'08 (Turku, Finland).

[71] Monteiro, P., Ropers, D., Mateescu, R., Freitas, A., de Jong, H., 2008. Temporal logic patterns for querying dynamic models of cellular interaction networks. Bioinformatics 24 (16), i227–i233.

[72] Regev, A., Shapiro, E., 2002. Cells as computation. Nature 419 (6905), 343.

[73] Ropers, D., de Jong, H., Page, M., Schneider, D., Geiselmann, J., 2006. Qualitative simulation of the carbon starvation response in *Escherichia coli*. Biosystems 84 (2), 124–152.

[74] Schoeberl, B., Eichler-Jonsson, C., Gilles, E.-D., Mller, G., 2002. Computational modeling of the dynamics of the MAP kinase cascade activated by surface and internalized EGF receptors. Nature Biotechnology 20 (4), 370–375.

[75] Sepulchre, J.-A., Reverchon, S., Nasser, W., 2007. Modeling the onset of virulence in a pectinolytic bacterium. J. Theor. Biol. 44 (2), 239–257.

[76] Shen, X., Collier, J., Dill, D., Shapiro, L., Horowitz, M., Mcadams, H., 2008. Architecture and inherent robustness of a bacterial cell-cycle control system. Proceedings of the National Academy of Sciences of the USA 105 (32), 11340–11345.

[77] Streett, R., 1982. Propositional dynamic logic of looping and converse. Information and Control (54), 121–141.

[78] Tarski, A., 1955. A lattice-theoretical fixpoint theorem and its applications. Pacific Journal of Mathematics 5, 285–309.

[79] Thomas, R., Kaufman, M., 2001. Multistationarity, the basis of cell differentiation and memory: I. Structural conditions of multistationarity and other nontrivial behavior. Chaos 11 (1), 170–179.

[80] Thomas, R., Thieffry, D., Kaufman, M., 1995. 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.

[81] Thomas, W., 1989. Computation Tree Logic and regular ω-languages. Vol. 354 of Lecture Notes in Computer Science. pp. 690–713.

[82] Tiwari, A., Khanna, G., 2002. Series abstractions for hybrid automata. In: Tomlin, C., Greenstreet, M. (Eds.), Proc. HSCC 2002. Vol. 2289 of LNCS. Springer-Verlag, pp. 465–478.

[83] Usseglio Viretta, A., Fussenegger, M., 2004. Modeling the quorum sensing regulatory network of human-pathogenic *Pseudomonas aeruginosa*. Biotechnol. Prog. 20 (3), 670–678.

[84] Vergauwen, B., Lewi, J., July 1994. Efficient local correctness checking for single and alternating boolean equation systems. In: Abiteboul, S., Shamir, E. (Eds.), Proceedings of the 21st ICALP (Vienna). Vol. 820 of Lecture Notes in Computer Science. Springer Verlag, Berlin, pp. 304–315.

[85] Wolper, P., 1982. A translation from full branching time temporal logic to one letter propositional dynamic logic with looping. Unpublished manuscript.

[86] Wolper, P., January-February 1983. Temporal logic can be more expressive. Information and Control 56 (1/2), 72–99.

A. Proofs of the translation from CTRL to MESs

A.1. Translation from CTRL to RESs

A few additional definitions and lemmas are required in order to prove Proposition 1. Given a propositional environment $\delta = [U_1/X_1, ..., U_n/X_n]$, its *support* is defined as $supp(\delta) = \{X_1, ..., X_n\}$, i.e., the set of variables that are mapped by δ to state sets. It is straightforward to show that, for environments with disjoint supports, the $\oslash$ operator is associative, commutative, and has the empty environment $[\,]$ as neutral element. Moreover, $supp([\![B]\!]_K\delta) = bv(B)$ and $supp([\![BL]\!]_K\delta) = bv(BL)$ for any Kripke structure K, equation block B, equation block list BL, and environment δ.

Lemma 2. *Let K be a Kripke structure, B an equation block, and δ_1, δ_2 two propositional environments such that $supp(\delta_1) \cap supp(\delta_2) = \emptyset$ and $fv(B) \subseteq supp(\delta_1)$. Then:*

$$[\![B]\!]_K(\delta_1 \oslash \delta_2) = [\![B]\!]_K\delta_1.$$

Proof Let $B = \{X_i \stackrel{\sigma}{=} \varphi_i\}_{1\leq i\leq n}$ be an equation block and δ_1, δ_2 two propositional environments as stated in the hypothesis. The semantics of B in the context of an environment δ is determined by the associated functional $\Phi_\delta : (2^S)^n \to (2^S)^n$ defined as follows:

$$\Phi_\delta(U_1, ..., U_n) = \langle [\![\varphi_i]\!]_K((\delta_1 \oslash \delta_2) \oslash [U_1/X_1, ..., U_n/X_n])\rangle_{1\leq i\leq n}$$

To prove the lemma, we show that the two functionals $\Phi_{(\delta_1 \oslash \delta_2)}$ and Φ_{δ_1} are identical, i.e., $[\![\varphi]\!]_K((\delta_1 \oslash \delta_2) \oslash [U_1/X_1, ..., U_n/X_n]) = [\![\varphi]\!]_K(\delta_1 \oslash [U_1/X_1, ..., U_n/X_n])$ for any formula φ and any $U_1, ..., U_n \subseteq S$. We proceed by structural induction on φ.

- $\varphi ::= p$:

  $[\![p]\!]_K((\delta_1 \oslash \delta_2) \oslash [U_1/X_1, ..., U_n/X_n])$
  $= \{s \in S \mid p \in L(s)\}$ by def. of $[\![\ ]\!]$
  $= [\![p]\!]_K(\delta_1 \oslash [U_1/X_1, ..., U_n/X_n])$ by def. of $[\![\ ]\!]$.

- $\varphi ::= X$:

  Two cases are possible.

  1. $X \in \{X_1, ..., X_n\}$, i.e., X is bound in B. Let $i \in [1, n]$ such that $X = X_i$.
     $[\![X_i]\!]_K((\delta_1 \oslash \delta_2) \oslash [U_1/X_1, ..., U_n/X_n])$
     $= ((\delta_1 \oslash \delta_2) \oslash [U_1/X_1, ..., U_n/X_n])(X_i)$ by def. of $[\![\ ]\!]$
     $= [U_1/X_1, ..., U_n/X_n](X_i)$ by def. of $\oslash$
     $= U_i$ by def. of $[\ ]$
     $= [\![X_i]\!]_K(\delta_1 \oslash [U_1/X_1, ..., U_n/X_n])$ by def. of $[\![\ ]\!]$.
  2. $X \notin \{X_1, ..., X_n\}$, i.e., X is free in B. This means $X \in supp(\delta_1)$.
     $[\![X]\!]_K((\delta_1 \oslash \delta_2) \oslash [U_1/X_1, ..., U_n/X_n])$
     $= ((\delta_1 \oslash \delta_2) \oslash [U_1/X_1, ..., U_n/X_n])(X)$ by def. of $[\![\ ]\!]$
     $= (\delta_1 \oslash \delta_2)(X)$ by def. of $\oslash$
     $= \delta_1(X)$ $fv(B) \not\subseteq supp(\delta_2)$
     $= [\![X]\!]_K\delta_1$ by def. of $[\![\ ]\!]$
     $= [\![X]\!]_K(\delta_1 \oslash [U_1/X_1, ..., U_n/X_n])$ by def. of $\oslash$.

- $\varphi ::= \varphi_1 \vee \varphi_2$ (similarly for $\varphi ::= \varphi_1 \wedge \varphi_2$):

  $[\![\varphi_1 \vee \varphi_2]\!]_K((\delta_1 \oslash \delta_2) \oslash [U_1/X_1, ..., U_n/X_n])$
  $= ([\![\varphi_1]\!]_K((\delta_1 \oslash \delta_2) \oslash [U_1/X_1, ..., U_n/X_n])) \cup$
  $([\![\varphi_2]\!]_K((\delta_1 \oslash \delta_2) \oslash [U_1/X_1, ..., U_n/X_n]))$ by def. of $[\![\ ]\!]$
  $= [\![\varphi_1]\!]_K(\delta_1 \oslash [U_1/X_1, ..., U_n/X_n]) \cup [\![\varphi_2]\!]_K((\delta_1 \oslash [U_1/X_1, ..., U_n/X_n])$
  by inductive hypothesis
  $= [\![\varphi_1 \vee \varphi_2]\!]_K(\delta_1 \oslash [U_1/X_1, ..., U_n/X_n])$ by def. of $[\![\ ]\!]$.

- $\varphi ::= \mathsf{EF}_\rho\varphi$ (similarly for $\varphi ::= \mathsf{AF}_\rho\varphi \mid \mathsf{EG}_\rho\varphi \mid \mathsf{AG}_\rho\varphi$):

$$
\begin{aligned}
&[\![\mathsf{EF}_\rho\varphi]\!]_K((\delta_1 \oslash \delta_2) \oslash [U_1/X_1, ..., U_n/X_n]) \\
&= \{s \in S \mid \exists \pi \in Path_K(s).\exists i \geq 0.\pi_{0,i} \models_K \rho \wedge \\
&\quad \pi_i \in [\![\varphi]\!]_K((\delta_1 \oslash \delta_2) \oslash [U_1/X_1, ..., U_n/X_n])\} \text{ by def. of } [\![\ ]\!] \\
&= \{s \in S \mid \exists \pi \in Path_K(s).\exists i \geq 0.\pi_{0,i} \models_K \rho \wedge \\
&\quad \pi_i \in [\![\varphi]\!]_K(\delta_1 \oslash [U_1/X_1, ..., U_n/X_n])\} \qquad \text{by inductive hypothesis} \\
&= [\![\mathsf{EF}_\rho\varphi]\!]_K(\delta_1 \oslash [U_1/X_1, ..., U_n/X_n]) \qquad \text{by def. of } [\![\ ]\!].
\end{aligned}
$$

- $\varphi ::= \mathsf{EF}^\infty_\rho$ (similarly for $\varphi ::= \mathsf{AF}^\infty_\rho \mid \mathsf{EG}^\dashv_\rho \mid \mathsf{AG}^\dashv_\rho$):

  $[\![\mathsf{EF}^\infty_\rho]\!]_K(\delta_1 \oslash \delta_2) = [\![\mathsf{EF}^\infty_\rho]\!]_K\delta_1$ because EF^∞_ρ is closed, so its interpretation is independent of any environment δ.

□

Lemma 3. *Let K be a Kripke structure and BL_1, BL_2 be two closed equation block lists. Then:*

$$[\![BL_1; BL_2]\!]_K = [\![BL_1]\!]_K \oslash [\![BL_2]\!]_K.$$

Proof Let K, BL_1, BL_2 as stated in the hypothesis. We proceed by structural induction on BL_1.

- $BL_1 ::= \varepsilon$:

$$
\begin{aligned}
[\![\varepsilon; BL_2]\!]_K &= [\![BL_2]\!]_K \qquad \text{by def. of ;} \\
&= [\ ] \oslash [\![BL_2]\!]_K \\
&= [\![\varepsilon]\!]_K \oslash [\![BL_2]\!]_K \quad \text{by def. of } [\![\ ]\!].
\end{aligned}
$$

- $BL_1 ::= B.BL_1$:

$$
\begin{aligned}
[\![(B.BL_1); BL_2]\!]_K &= [\![B.(BL_1; BL_2)]\!]_K \\
&\qquad \text{by def. of ;} \\
&= [\![B]\!]_K([\![BL_1; BL_2]\!]_K) \oslash [\![BL_1; BL_2]\!]_K \\
&\qquad \text{by def. of } [\![\ ]\!] \\
&= [\![B]\!]_K([\![BL_1]\!]_K \oslash [\![BL_2]\!]_K) \oslash ([\![BL_1]\!]_K \oslash [\![BL_2]\!]_K) \\
&\qquad \text{by ind. hyp.} \\
&= [\![B]\!]_K([\![BL_1]\!]_K) \oslash ([\![BL_1]\!]_K \oslash [\![BL_2]\!]_K) \\
&\qquad \text{by Lemma 2} \\
&= ([\![B]\!]_K([\![BL_1]\!]_K) \oslash [\![BL_1]\!]_K) \oslash [\![BL_2]\!]_K \\
&\qquad \text{by assoc.} \\
&= [\![B.BL_1]\!]_K \oslash [\![BL_2]\!]_K \\
&\qquad \text{by def. of } [\![\ ]\!].
\end{aligned}
$$

□

Proof *(Proposition 1).* Let K be a Kripke structure, φ be a state formula of CTRL, and δ be a propositional environment. We proceed by structural induction on φ.

- $\varphi ::= p$:

$$\begin{aligned}
[\![t(p)]\!]_K\delta &= [\![\langle X, \{X \stackrel{\mu}{=} p\}\rangle]\!]_K\delta && \text{by def. of } t\\
&= ([\![\{X \stackrel{\mu}{=} p\}]\!]_K\delta)(X) && \text{by def. of } [\![\ ]\!]\\
&= [\![p]\!]_K\delta && \text{by def. of } [\![\ ]\!].
\end{aligned}$$

- $\varphi ::= \varphi_1 \vee \varphi_2$ (similarly for $\varphi ::= \varphi_1 \wedge \varphi_2$):

$$\begin{aligned}
[\![t(\varphi_1 \vee \varphi_2)]\!]_K\delta &= [\![\langle X, \{X \stackrel{\mu}{=} t_X(\varphi_1) \vee t_X(\varphi_2)\}.(t_{BL}(\varphi_1); t_{BL}(\varphi_2))\rangle]\!]_K\delta\\
&\qquad \text{by def. of } t\\
&= ([\![\{X \stackrel{\mu}{=} t_X(\varphi_1) \vee t_X(\varphi_2)\}.(t_{BL}(\varphi_1); t_{BL}(\varphi_2))]\!]_K\delta)(X)\\
&\qquad \text{by def. of } [\![\ ]\!]\\
&= ([\![\{X \stackrel{\mu}{=} t_X(\varphi_1) \vee t_X(\varphi_2)\}]\!]_K(\delta \oslash [\![t_{BL}(\varphi_1); t_{BL}(\varphi_2)]\!]_K\delta)\\
&\quad \oslash([\![t_{BL}(\varphi_1); t_{BL}(\varphi_2)]\!]_K\delta))(X)\\
&\qquad \text{by def. of } [\![\ ]\!]\\
&= ([\![\{X \stackrel{\mu}{=} t_X(\varphi_1) \vee t_X(\varphi_2)\}]\!]_K(\delta \oslash [\![t_{BL}(\varphi_1); t_{BL}(\varphi_2)]\!]_K\delta))(X)\\
&= [\![t_X(\varphi_1)]\!]_K(\delta \oslash [\![t_{BL}(\varphi_1)]\!]_K\delta \oslash [\![t_{BL}(\varphi_2)]\!]_K\delta) \cup\\
&\quad [\![t_X(\varphi_2)]\!]_K(\delta \oslash [\![t_{BL}(\varphi_1)]\!]_K\delta \oslash [\![t_{BL}(\varphi_2)]\!]_K\delta)\\
&= [\![t_X(\varphi_1)]\!]_K([\![t_{BL}(\varphi_1)]\!]_K\delta) \cup [\![t_X(\varphi_2)]\!]_K([\![t_{BL}(\varphi_2)]\!]_K\delta)\\
&\qquad \text{by Lemma 2}\\
&= [\![t(\varphi_1)]\!]_K\delta \cup [\![t(\varphi_2)]\!]_K\delta\\
&\qquad \text{by def. of } t\\
&= [\![\varphi_1]\!]_K\delta \cup [\![\varphi_2]\!]_K\delta\\
&\qquad \text{by ind. hyp.}\\
&= [\![\varphi_1 \vee \varphi_2]\!]_K\delta\\
&\qquad \text{by def. of } [\![\ ]\!].
\end{aligned}$$

- $\varphi ::= \mathsf{EF}_\rho\varphi$ (similarly for $\varphi ::= \mathsf{AF}_\rho\varphi \mid \mathsf{EG}_\rho\varphi \mid \mathsf{AG}_\rho\varphi$):

$$
\begin{aligned}
[\![t(\mathsf{EF}_\rho\varphi)]\!]_K\delta &= [\![\langle X, \{X \stackrel{\mu}{=} \mathsf{EF}_\rho t_X(\varphi)\}.t_{BL}(\varphi)\rangle]\!]_K\delta \\
&\qquad \text{by def. of } t \\
&= ([\![t_{BL}(\varphi)]\!]_K\delta \oslash [\![\{X \stackrel{\mu}{=} \mathsf{EF}_\rho t_X(\varphi)\}]\!]_K([\![t_{BL}(\varphi)]\!]_K\delta))(X) \\
&\qquad \text{by def. of } [\![\ ]\!] \\
&= ([\![\{X \stackrel{\mu}{=} \mathsf{EF}_\rho t_X(\varphi)\}]\!]_K([\![t_{BL}(\varphi)]\!]_K\delta))(X) \\
&= [\![\{\mathsf{EF}_\rho t_X(\varphi)\}]\!]_K([\![t_{BL}(\varphi)]\!]_K\delta) \\
&\qquad \text{by def. of } [\![\ ]\!] \\
&= \{s \in S \mid \exists \pi \in Path_K(s).\exists i \geq 0.\pi_{0,i} \models_K \rho \wedge \\
&\quad \pi_i \in [\![t_X(\varphi)]\!]_K([\![t_{BL}(\varphi)]\!]_K\delta)\} \\
&\qquad \text{by def. of } [\![\ ]\!] \\
&= \{s \in S \mid \exists \pi \in Path_K(s).\exists i \geq 0.\pi_{0,i} \models_K \rho \wedge \\
&\quad \pi_i \in [\![\varphi]\!]_K\delta\} \\
&\qquad \text{by ind. hyp.} \\
&= [\![\mathsf{EF}_\rho\varphi]\!]_K\delta \\
&\qquad \text{by def. of } [\![\ ]\!].
\end{aligned}
$$

- $\varphi ::= \mathsf{EF}^\infty_\rho$ (similarly for $\varphi ::= \mathsf{AG}^\dashv_\rho$):

  $$
  \begin{aligned}
  [\![t(\mathsf{EF}^\infty_\rho)]\!]_K\delta &= [\![\langle X, \{X \stackrel{\nu}{=} \mathsf{EF}_\rho X\}\rangle]\!]_K \qquad \text{by def. of } t. \\
  &= ([\![\{X \stackrel{\nu}{=} \mathsf{EF}_\rho X\}]\!]_K)(X) = \nu\Phi \quad \text{by def. of } [\![\ ]\!],
  \end{aligned}
  $$

  where $\Phi : 2^S \to 2^S$, $\Phi(U) = [\![\mathsf{EF}_\rho X]\!]_K[U/X]$. Note that the δ environment is omitted in the definition of Φ because the equation block $\{X \stackrel{\nu}{=} \mathsf{EF}_\rho X\}$ is closed.

  The lattice $\langle 2^S, \emptyset, S, \cap, \cup\rangle$ being finite, the maximal fixed point $\nu\Phi$ has also the following iterative characterization [58]:

  $$\nu\Phi = \bigcap_{j\geq 0} \Phi^j(S), \quad \text{where} \quad \Phi^0(S) = S, \quad \Phi^j(S) = [\![\mathsf{EF}_\rho X]\!]_K[\Phi^{j-1}(S)/X].$$

  Intuitively, the terms $\Phi^j(S)$ contain those states from which there is an outoing sequence having a prefix that matches ρ^j:

  $$\Phi^j(S) = \{s \in S \mid \exists \pi \in Path_K(s).\exists i \geq 0.\pi_{0,i} \models_K \rho^j\}$$

  This can be easily shown by induction on j. For $j = 0$, we take $i = 0$ (empty prefix). For the inductive step, we have:

$$\begin{aligned}\Phi^{j+1}(S) &= \{s \in S \mid \exists \pi \in Path_K(s).\exists i \geq 0.\pi_{0,i} \models_K \rho \wedge \pi_i \in \Phi^j(S)\} \\ &\qquad \text{by def. of } \Phi \\ &= \{s \in S \mid \exists \pi \in Path_K(s).\exists i \geq 0. \\ &\qquad\qquad \pi_{0,i} \models_K \rho \wedge \exists \pi' \in Path_K(\pi_i).\exists l \geq 0.\pi'_{0,l} \models_K \rho^j\} \\ &\qquad \text{by ind. hyp.} \\ &= \{s \in S \mid \exists \pi \in Path_K(s).\exists i \geq 0.\pi_{0,i} \models_K \rho^{j+1}\} \\ &\qquad \text{repl. } i \text{ by } i+l.\end{aligned}$$

To show that $[\![\mathsf{EF}^\infty_\rho]\!]_K \subseteq \nu\Phi$, let $s \in [\![\mathsf{EF}^\infty_\rho]\!]_K$ and $j \geq 0$. From the definition of EF^∞_ρ, there exists $\pi \in Path_K(s)$ and $i \geq 0$ such that $\pi_{0,i} \models_K \rho^j$, which implies $s \in \Phi^j(S)$. Since this holds for every $j \geq 0$, it means that $s \in \bigcap_{j\geq 0} \Phi^j(S)$, i.e., $s \in \nu\Phi$.

To show that $\nu\Phi \subseteq [\![\mathsf{EF}^\infty_\rho]\!]_K$, let $s \in \nu\Phi$. Since $\nu\Phi$ is a fixed point of Φ, we have:

$$\begin{aligned}\nu\Phi &= \Phi(\nu\Phi) = [\![\mathsf{EF}_\rho X]\!]_K[\nu\Phi/X] \\ &= \{s \in S \mid \exists \pi \in Path_K(s).\exists i \geq 0.\pi_{0,i} \models_K \rho \wedge \pi_i \in \nu\Phi\}.\end{aligned}$$

Based on this, we construct the following path:

$$\overline{\pi} = \overline{\pi}_{i_0} \rightarrow \cdots \rightarrow \overline{\pi}_{i_1} \rightarrow \cdots \rightarrow \overline{\pi}_{i_2} \rightarrow \cdots \rightarrow \overline{\pi}_{i_j} \cdots$$

where $\overline{\pi}_{i_j} \in \nu\Phi$ for every $j \geq 0$, $i_0 = 0$, $\overline{\pi}_{i_0} = s$, and the intervals $\overline{\pi}_{i_j} \rightarrow \cdots \rightarrow \overline{\pi}_m \rightarrow \cdots \rightarrow \overline{\pi}_{i_{j+1}}$ are defined as follows. Since $\overline{\pi}_{i_j} \in \nu\Phi$, according to the equation above, there exists $\pi \in Path_K(\overline{\pi}_{i_j})$ and $l \geq 0$ such that $\pi_{0,l} \models_K \rho$ and $\pi_l \in \nu\Phi$. We take $i_{j+1} = i_j + l$ and for each $m \in [i_j, i_{j+1}]$, $\overline{\pi}_m = \pi_{m-i_j}$. The infinite path $\overline{\pi}$ is such that for every $j \geq 0$, there exists $i' = i_j$ such that $\overline{\pi}_{0,i'} \models_K \rho^j$, and therefore $s \in [\![\mathsf{EF}^\infty_\rho]\!]_K$.

□

A.2. Translation from RESs to MESs

Some additional definitions and lemmas are needed in order to prove the translation. Let $B = \{X_i \stackrel{\sigma}{=} \varphi_i\}_{1\leq i\leq n}$ be an equation block and $\Phi_\delta : (2^S)^n \rightarrow (2^S)^n$, $\Phi_\delta(U_1, ..., U_n) = \langle [\![\varphi_i]\!]_K(\delta \oslash [U_1/X_1, ..., U_n/X_n])\rangle_{1\leq i\leq n}$ be its associated functional in the context of a Kripke structure K and an environment δ. For a given $l \in [1, n]$, the *projection* of Φ_δ on the equations $[l, n]$, noted $\Phi^{l,n} : (2^S)^{n-l+1} \rightarrow (2^S)^{n-l+1}$, is defined as follows: $\Phi^{l,n}_\delta(U_l, ..., U_n) = \langle [\![\varphi_j]\!]_K(\delta \oslash [U_l/X_l, ..., U_n/X_n])\rangle_{l\leq j\leq n}$. Similarly, the projection of a value $\langle U_1, ..., U_n\rangle \in (2^S)^n$ on the fields $[l, n]$ is defined as $\langle U_1, ..., U_n\rangle_{[l,n]} = \langle U_l, ..., U_n\rangle$.

A.2.1. Operators EF_ρ *and* AG_ρ

Lemma 4. *Let* $B = \{X_i \stackrel{\sigma}{=} \varphi_i\}_{1\leq i\leq n}$ *be an equation block,* K *be a Kripke structure,* δ *be an environment, and* $\Phi_\delta : (2^S)^n \to (2^S)^n$ *be the functional associated to* B*,* K*, and* δ*. Then, for all* $l \in [1, n]$*:*

$$\sigma\Phi^{l,n}_{\delta\oslash[(\sigma\Phi_\delta)_1/X_1,...,(\sigma\Phi_\delta)_{l-1}/X_{l-1}]} = \langle(\sigma\Phi_\delta)_l, ..., (\sigma\Phi_\delta)_n\rangle$$

where $\Phi^{l,n}_\delta : (2^S)^{n-l+1} \to (2^S)^{n-l+1}$ *is the projection of* Φ_δ *on the equations* $[l, n]$*.*

Proof Let B, K, δ, and l as stated in the hypothesis. We show the equality by double inclusion, only for $\sigma = \mu$, the proof for the case $\sigma = \nu$ being symmetric.
Inclusion "$\sqsupseteq$": By definition of fixed points we have $\mu\Phi_\delta = \Phi_\delta(\mu\Phi_\delta)$, meaning that for all $l \leq j \leq n$:

$$(\mu\Phi_\delta)_j = [\![\varphi_j]\!]_K(\delta \oslash [(\mu\Phi_\delta)_1/X_1, ..., (\mu\Phi_\delta)_n/X_n]) =$$
$$[\![\varphi_j]\!]_K((\delta \oslash [(\mu\Phi_\delta)_1/X_1, ..., (\mu\Phi_\delta)_{l-1}/X_{l-1}]) \oslash [(\mu\Phi_\delta)_l/X_l, ..., (\mu\Phi_\delta)_n/X_n])$$

This in turn means that:

$$\Phi^{l,n}_{\delta\oslash[(\sigma\Phi_\delta)_1/X_1,...,(\sigma\Phi_\delta)_{l-1}/X_{l-1}]}((\mu\Phi_\delta)_l, ..., (\mu\Phi_\delta)_n) = \langle(\mu\Phi_\delta)_l, ..., (\mu\Phi_\delta)_n\rangle$$

i.e., $\langle(\mu\Phi_\delta)_l, ..., (\mu\Phi_\delta)_n\rangle$ is a fixed point of $\Phi^{l,n}_{\delta\oslash[(\sigma\Phi_\delta)_1/X_1,...,(\sigma\Phi_\delta)_{l-1}/X_{l-1}]}$, and therefore it is greater than the least fixed point of this functional:

$$\mu\Phi^{l,n}_{\delta\oslash[(\sigma\Phi_\delta)_1/X_1,...,(\sigma\Phi_\delta)_{l-1}/X_{l-1}]} \sqsubseteq \langle(\mu\Phi_\delta)_l, ..., (\mu\Phi_\delta)_n\rangle.$$

Inclusion "$\sqsupseteq$": We use the iterative characterization [58] of $\mu\Phi_\delta$ on the finite lattice $\langle 2^{S^n}, \emptyset, S^n, \sqcap, \sqcup\rangle$ (the operations $\sqcap$ and $\sqcup$ are the pairwise extensions of $\cap$ and $\cup$):

$$\mu\Phi_\delta = \bigcup_{k\geq 0} \Phi^k_\delta(\emptyset^n), \quad \text{where} \quad \Phi^0_\delta(\emptyset^n) = \emptyset^n, \quad \Phi^{k+1}_\delta(\emptyset^n) = \Phi_\delta(\Phi^k_\delta(\emptyset^n)).$$

We show, by induction on k, that $(\Phi^k_\delta(\emptyset^n))_{[l,n]} \sqsubseteq \mu\Phi^{l,n}_{\delta\oslash[(\sigma\Phi_\delta)_1/X_1,...,(\sigma\Phi_\delta)_{l-1}/X_{l-1}]}$.
Base step. $(\Phi^0_\delta(\emptyset^n))_{[l,n]} = (\emptyset^n)_{[l,n]} = \emptyset^{n-l+1} \sqsubseteq \mu\Phi^{l,n}_{\delta\oslash[(\sigma\Phi_\delta)_1/X_1,...,(\sigma\Phi_\delta)_{l-1}/X_{l-1}]}$.
Inductive step. We have:

$$
\begin{aligned}
(\Phi_\delta^{k+1}(\emptyset^n))_{[l,n]} &= (\Phi_\delta(\Phi_\delta^k(\emptyset^n)))_{[l,n]} \\
&= \langle [\![\varphi_j]\!]_K(\delta \oslash [(\Phi_\delta^k(\emptyset^n))_1/X_1, ..., (\Phi_\delta^k(\emptyset^n))_n/X_n])\rangle_{l\le j\le n} \\
&\qquad \text{by def. of } \Phi \\
&= \langle [\![\varphi_j]\!]_K((\delta \oslash [(\Phi_\delta^k(\emptyset^n))_1/X_1, ..., (\Phi_\delta^k(\emptyset^n))_{l-1}/X_{l-1}]) \oslash \\
&\qquad [(\Phi_\delta^k(\emptyset^n))_l/X_l, ..., (\Phi_\delta^k(\emptyset^n))_n/X_n])\rangle_{l\le j\le n} \\
&\sqsubseteq \langle [\![\varphi_j]\!]_K((\delta \oslash [(\mu\Phi_\delta)_1/X_1, ..., (\mu\Phi_\delta)_{l-1}/X_{l-1}]) \oslash \\
&\qquad [(\Phi_\delta^k(\emptyset^n))_l/X_l, ..., (\Phi_\delta^k(\emptyset^n))_n/X_n])\rangle_{l\le j\le n} \\
&\qquad \text{by monotonicity} \\
&= \Phi^{l,n}_{\delta\oslash[(\mu\Phi_\delta)_1/X_1,...,(\mu\Phi_\delta)_{l-1}/X_{l-1}]}((\Phi_\delta^k(\emptyset^n))_l, ..., (\Phi_\delta^k(\emptyset^n))_n) \\
&\qquad \text{by def. of } \Phi^{l,n} \\
&= \Phi^{l,n}_{\delta\oslash[(\mu\Phi_\delta)_1/X_1,...,(\mu\Phi_\delta)_{l-1}/X_{l-1}]}((\Phi_\delta^k(\emptyset^n))_{[l,n]}) \\
&\sqsubseteq \Phi^{l,n}_{\delta\oslash[(\mu\Phi_\delta)_1/X_1,...,(\mu\Phi_\delta)_{l-1}/X_{l-1}]}(\mu\Phi^{l,n}_{\delta\oslash[(\sigma\Phi_\delta)_1/X_1,...,(\sigma\Phi_\delta)_{l-1}/X_{l-1}]}) \\
&\qquad \text{by ind. hyp.} \\
&= \mu\Phi^{l,n}_{\delta\oslash[(\sigma\Phi_\delta)_1/X_1,...,(\sigma\Phi_\delta)_{l-1}/X_{l-1}]} \\
&\qquad \text{by def. of } \mu.
\end{aligned}
$$

Thus, $(\mu\Phi_\delta)_{[l,n]} = (\bigcup_{k\ge 0}\Phi_\delta^k(\emptyset^n))_{[l,n]} = \bigcup_{k\ge 0}(\Phi_\delta^k(\emptyset^n))_{[l,n]} \sqsubseteq \mu\Phi^{l,n}_{\delta\oslash[(\sigma\Phi_\delta)_1/X_1,...,(\sigma\Phi_\delta)_{l-1}/X_{l-1}]}$, which concludes the proof. □

The following lemma allows to replace an equation of a block by a set of equations, provided that the interpretation of the variable in the left-hand side of the equation remains unchanged in the original and the substituting block w.r.t. all environments.

Lemma 5 (Substitution). *Let $B = \{X_i \stackrel{\sigma}{=} \varphi_i\}_{1\le i\le n}$ be an equation block, and let $\{X_n \stackrel{\sigma}{=} \psi_n, Y_j \stackrel{\sigma}{=} \psi_j\}_{n<j\le m}$ be another block suitable for the substitution of the equation $X_n \stackrel{\sigma}{=} \varphi_n$ such that $([\![\{X_n \stackrel{\sigma}{=} \varphi_n\}]\!]_K\delta)(X_n) = ([\![\{X_n \stackrel{\sigma}{=} \psi_n, Y_j \stackrel{\sigma}{=} \psi_j\}_{n<j\le m}]\!]_K\delta)(X_n)$ for any Kripke structure K and environment δ. Then:*

$$
\begin{aligned}
&([\![\{X_i \stackrel{\sigma}{=} \varphi_i\}_{1\le i\le n}[X_n \stackrel{\sigma}{=} \varphi_n := X_n \stackrel{\sigma}{=} \psi_n, Y_j \stackrel{\sigma}{=} \psi_j]_{n<j\le m}]\!]_K\delta)(X_i) = \\
&([\![\{X_i \stackrel{\sigma}{=} \varphi_i\}_{1\le i\le n}]\!]_K\delta)(X_i)
\end{aligned}
$$

for all $i \in [1, n]$ and for any K, δ.

Proof We show the lemma for $\sigma = \mu$, the proof for the case $\sigma = \nu$ being symmetric. Let $\Phi_\delta^{1,m} : (2^S)^m \to (2^S)^m$ be the functional associated to the substituted equation block, defined as follows:

$$
\begin{aligned}
&\Phi_\delta^{1,m}(U_1, ..., U_n, W_{n+1}, ..., W_n) = \\
&\qquad \langle [\![\varphi_i]\!]_K(\delta \oslash [U_1/X_1, ..., U_n/X_n, W_{n+1}/Y_{n+1}, ..., W_m/Y_m]), \\
&\qquad\ [\![\psi_j]\!]_K(\delta \oslash [U_1/X_1, ..., U_n/X_n, W_{n+1}/Y_{n+1}, ..., W_m/Y_m])\rangle_{1\le i<n, n\le j\le m}
\end{aligned}
$$

We first show that $\langle(\mu\Phi_\delta^{1,m})_1, ..., (\mu\Phi_\delta^{1,m})_n\rangle$ is a fixed point of the functional Φ_δ associated to B and δ. From the definition of $\mu\Phi_\delta^{1,m}$, it follows that $[\![\varphi_i]\!]_K(\delta \oslash [(\mu\Phi_\delta^{1,m})_1/X_1, ..., (\mu\Phi_\delta^{1,m})_n/X_n, (\mu\Phi_\delta^{1,m})_{n+1}/Y_{n+1}, ..., (\mu\Phi_\delta^{1,m})_m/Y_m]) = (\mu\Phi_\delta^{1,m})_i$ for all $i \in [1, n-1]$. The suitability condition $\bigcup_{i=1}^n fv(\varphi_i) \cap \{Y_{n+1}, ..., Y_m\} = \emptyset$ implies that all formulas φ_i for $i \in [1, n-1]$ depend only upon $X_1, ..., X_n$ and therefore $[\![\varphi_i]\!]_K(\delta \oslash [(\mu\Phi_\delta^{1,m})_1/X_1, ..., (\mu\Phi_\delta^{1,m})_n/X_n]) = (\mu\Phi_\delta^{1,m})_i$. To show that this equality also holds for $i = n$, we apply Lemma 4 for $l = n$ on the substituted block and we obtain:

$$\mu\Phi^{n,m}_{\delta \oslash [(\mu\Phi_\delta^{1,m})_1/X_1, ..., (\mu\Phi_\delta^{1,m})_{n-1}/X_{n-1}]} = \langle(\mu\Phi_\delta^{1,m})_n, ..., (\mu\Phi_\delta^{1,m})_m\rangle$$

where $\Phi_\delta^{n,m} : (2^S)^{m-n+1} \to (2^S)^{m-n+1}$ is the projection of $\Phi_\delta^{1,m}$ on the equations $[n, m]$. From the hypothesis of the lemma and the definition of the interpretation $[\![\{X_n \stackrel{\sigma}{=} \psi_n, Y_j \stackrel{\sigma}{=} \psi_j\}_{n<j\leq m}]\!]_K\delta$, this implies:

$$([\![\{X_n \stackrel{\sigma}{=} \varphi_n\}]\!]_K(\delta \oslash [(\mu\Phi_\delta^{1,m})_1/X_1, ..., (\mu\Phi_\delta^{1,m})_{n-1}/X_{n-1}]))(X_n) = (\mu\Phi_\delta^{1,m})_n$$

or, according to the definition of $[\![\{X_n \stackrel{\sigma}{=} \varphi_n\}]\!]_K\delta$:

$$\mu\Phi^{n}_{\delta \oslash [(\mu\Phi_\delta^{1,m})_1/X_1, ..., (\mu\Phi_\delta^{1,m})_{n-1}/X_{n-1}]} = (\mu\Phi_\delta^{1,m})_n$$

where $\Phi_\delta^n : 2^S \to 2^S$, $\Phi_\delta^n(U) = [\![\varphi_n]\!]_K(\delta \oslash [U/X_n])$. Since $(\mu\Phi_\delta^{1,m})_n$ is by definition a fixed point of $\Phi^{n}_{\delta \oslash [(\mu\Phi_\delta^{1,m})_1/X_1, ..., (\mu\Phi_\delta^{1,m})_{n-1}/X_{n-1}]}$, this means:

$$([\![\varphi_n]\!]_K((\delta \oslash [(\mu\Phi_\delta^{1,m})_1/X_1, ..., (\mu\Phi_\delta^{1,m})_{n-1}/X_{n-1}]) \oslash [(\mu\Phi_\delta^{1,m})_n/X_n]))((\mu\Phi_\delta^{1,m})_n) = (\mu\Phi_\delta^{1,m})_n$$

i.e.,

$$([\![\varphi_n]\!]_K(\delta \oslash [(\mu\Phi_\delta^{1,m})_1/X_1, ..., (\mu\Phi_\delta^{1,m})_n/X_n]))((\mu\Phi_\delta^{1,m})_n) = (\mu\Phi_\delta^{1,m})_n.$$

Therefore, $\langle(\mu\Phi_\delta^{1,m})_i\rangle_{1\leq i\leq n}$ is a fixed point of Φ_δ.

It remains to show that this is indeed the minimal fixed point of Φ_δ. Since the lattice $\langle 2^{S^m}, \emptyset, S^m, \sqcap, \sqcup\rangle$ is finite (the operations $\sqcap$ and $\sqcup$ being the pairwise extensions of $\cap$ and $\cup$), the minimal fixed point $\mu\Phi_\delta^{1,m}$ also has an iterative characterization [58]:

$$\mu\Phi_\delta^{1,m} = \bigcup_{k\geq 0} (\Phi_\delta^{1,m})^k(\emptyset^m)$$

where $(\Phi_\delta^{1,m})^0(\emptyset^m) = \emptyset^m$, $(\Phi_\delta^{1,m})^{k+1}(\emptyset^m) = \Phi_\delta^{1,m}((\Phi_\delta^{1,m})^k(\emptyset^m))$.

We show, by induction on k, that $((\Phi_\delta^{1,m})^k(\emptyset^m))_i \subseteq (\mu\Phi_\delta)_i$ for all $i \in [1,n]$ and $k \geq 0$. Let $i \in [1,n]$.

Base step. $((\Phi_\delta^{1,m})^0(\emptyset^m))_i = \emptyset \subseteq (\mu\Phi_\delta)_i$.

Inductive step. For $i \in [1, n-1]$, we have:

$$\begin{aligned}
((\Phi_\delta^{1,m})^{k+1}(\emptyset^m))_i &= (\Phi_\delta^{1,m}((\Phi_\delta^{1,m})^k(\emptyset^m)))_i \\
&= [\![\varphi_i]\!]_K(\delta \oslash [((\Phi_\delta^{1,m})^k(\emptyset^m))_1/X_1, ..., ((\Phi_\delta^{1,m})^k(\emptyset^m))_m/Y_m]) \\
&\qquad \text{by def. of } [\![\ ]\!] \\
&= [\![\varphi_i]\!]_K(\delta \oslash [((\Phi_\delta^{1,m})^k(\emptyset^m))_1/X_1, ..., ((\Phi_\delta^{1,m})^k(\emptyset^m))_n/X_n]) \\
&\qquad \text{by suitability} \\
&\subseteq [\![\varphi_i]\!]_K(\delta \oslash [(\mu\Phi_\delta)_1/X_1, ..., (\mu\Phi_\delta)_n/X_n]) \\
&\qquad \text{by ind. hyp.} \\
&= (\mu\Phi_\delta)_i \\
&\qquad \text{by def. of } \mu\Phi_\delta.
\end{aligned}$$

For $i = n$, we have:

$$\begin{aligned}
((\Phi_\delta^{1,m})^{k+1}(\emptyset^m))_n &= [\![\psi_n]\!]_K(\delta \oslash [((\Phi_\delta^{1,m})^k(\emptyset^m))_1/X_1, ..., ((\Phi_\delta^{1,m})^k(\emptyset^m))_m/Y_m]) \\
&\subseteq [\![\psi_n]\!]_K(\delta \oslash [((\Phi_\delta^{1,m})^k(\emptyset^m))_1/X_1, ..., ((\Phi_\delta^{1,m})^k(\emptyset^m))_{n-1}/X_{n-1}, \\
&\qquad\qquad (\mu\Phi_\delta^{1,m})_n/X_n, ..., (\mu\Phi_\delta^{1,m})_m/Y_m]) \\
&\qquad \text{by def. } \mu\Phi_\delta^{1,m} \\
&= ([\![\{X_n \stackrel{\mu}{=} \psi_n, Y_j \stackrel{\mu}{=} \psi_j\}_{n<j\leq m}]\!]_K \\
&\quad (\delta \oslash [((\Phi_\delta^{1,m})^k(\emptyset^m))_1/X_1, ..., ((\Phi_\delta^{1,m})^k(\emptyset^m))_{n-1}/X_{n-1}]))(X_n) \\
&\qquad \text{by def. of } [\![\ ]\!] \\
&= ([\![X_n \stackrel{\mu}{=} \varphi_n]\!]_K \\
&\quad (\delta \oslash [((\Phi_\delta^{1,m})^k(\emptyset^m))_1/X_1, ..., ((\Phi_\delta^{1,m})^k(\emptyset^m))_{n-1}/X_{n-1}]))(X_n) \\
&\qquad \text{by hyp.} \\
&= \mu\Phi^n_{\delta \oslash [((\Phi_\delta^{1,m})^k(\emptyset^m))_1/X_1, ..., ((\Phi_\delta^{1,m})^k(\emptyset^m))_{n-1}/X_{n-1}]} \\
&\qquad \text{by def. of } [\![\ ]\!] \\
&\subseteq \mu\Phi^n_{\delta \oslash [(\mu\Phi_\delta)_1/X_1, ..., (\mu\Phi_\delta)_{n-1}/X_{n-1}]} \\
&\qquad \text{by ind. hyp.} \\
&= (\mu\Phi_\delta)_n \\
&\qquad \text{by Lemma 4.}
\end{aligned}$$

The last application of Lemma 4 above considers the block $B = \{X_i \stackrel{\sigma}{=} \varphi_i\}_{1\leq i\leq n}$ and takes $l = n$. This concludes the proof that $\langle(\mu\Phi_\delta^{1,m})_i\rangle_{1\leq i\leq n}$ is the least fixed point of Φ_δ. □

Lemma 5 allows to prove the correctness of a substitution by focusing only on the equations involved in the substitution, as illustrated in the proof

below.

Proof *(Proposition 2).* Let K be a Kripke structure, $B_1 = \{X_i \stackrel{\mu}{=} \varphi_i\}_{1 \leq i \leq n}$ and $B_2 = \{X_i \stackrel{\nu}{=} \varphi_i\}_{1 \leq i \leq n}$ two equation blocks, and δ a propositional environment as stated in the hypothesis. We show the proposition only for blocks of type B_1 and the substitutions in the upper part of Figure 5, the other cases being dual.

- Substitution $X \stackrel{\mu}{=} \mathsf{EF}_{\rho_1.\rho_2} Y := X \stackrel{\mu}{=} \mathsf{EF}_{\rho_1} Z, Z \stackrel{\mu}{=} \mathsf{EF}_{\rho_2} Y$. It is sufficient to show that this substitution satisfies the condition in the hypothesis of Lemma 5:

  $$(\llbracket \{X \stackrel{\mu}{=} \mathsf{EF}_{\rho_1.\rho_2} Y\} \rrbracket_K \delta)(X) = (\llbracket \{X \stackrel{\mu}{=} \mathsf{EF}_{\rho_1} Z, Z \stackrel{\mu}{=} \mathsf{EF}_{\rho_2} Y\} \rrbracket_K \delta)(X).$$

  By applying the definition of $\llbracket\ \rrbracket$ and simple properties about substitution of variables in a Res, we obtain:

  $$\begin{aligned}
  &(\llbracket \{X \stackrel{\mu}{=} \mathsf{EF}_{\rho_1.\rho_2} Y\} \rrbracket_K \delta)(X) \\
  &= \llbracket \mathsf{EF}_{\rho_1.\rho_2} Y \rrbracket_K \delta \\
  &= \{s \in S \mid \exists \pi \in Path_K(s). \exists i \geq 0. \pi_{0,i} \models_K \rho_1.\rho_2 \wedge \pi_i \in \llbracket Y \rrbracket_K\} \\
  &= \{s \in S \mid \exists \pi \in Path_K(s). \exists i \geq 0. \exists k \in [0,i]. \\
  &\qquad \pi_{0,k} \models_K \rho_1 \wedge \pi_{k,i} \models_K \rho_2 \wedge \pi_i \in \llbracket Y \rrbracket_K\} \\
  &= \{s \in S \mid \exists \pi \in Path_K(s). \exists k \geq 0. \\
  &\qquad \pi_{0,k} \models_K \rho_1 \wedge \exists i \geq k. \pi_{k,i} \models_K \rho_2 \wedge \pi_i \in \llbracket Y \rrbracket_K\} \\
  &= \{s \in S \mid \exists \pi \in Path_K(s). \exists k \geq 0. \pi_{0,k} \models_K \rho_1 \wedge \exists \pi' \in Path_K(\pi_k). \exists i \geq k. \\
  &\qquad \pi'_{k,i} \models_K \rho_2 \wedge \pi'_i \in \llbracket Y \rrbracket_K\} \\
  &= \{s \in S \mid \exists \pi \in Path_K(s). \exists k \geq 0. \pi_{0,k} \models_K \rho \wedge \pi_k \in \llbracket \mathsf{EF}_{\rho_2} Y \rrbracket_K\} \\
  &= \llbracket \mathsf{EF}_{\rho_1} \mathsf{EF}_{\rho_2} Y \rrbracket_K \delta \\
  &= (\llbracket \{X \stackrel{\mu}{=} \mathsf{EF}_{\rho_1} \mathsf{EF}_{\rho_2} Y\} \rrbracket_K \delta)(X) \\
  &= (\llbracket \{X \stackrel{\mu}{=} \mathsf{EF}_{\rho_1} Z, Z \stackrel{\mu}{=} \mathsf{EF}_{\rho_2} Y\} \rrbracket_K \delta)(X).
  \end{aligned}$$

- Substitution $X \stackrel{\mu}{=} \mathsf{EF}_{\rho_1 | \rho_2} Y := X \stackrel{\mu}{=} Z \vee U, Z \stackrel{\mu}{=} \mathsf{EF}_{\rho_1} Y, U \stackrel{\mu}{=} \mathsf{EF}_{\rho_2} Y$. As above, it is sufficient to show that:

  $$\begin{aligned}
  &(\llbracket \{X \stackrel{\mu}{=} \mathsf{EF}_{\rho_1 | \rho_2} Y\} \rrbracket_K \delta)(X) = \\
  &(\llbracket \{X \stackrel{\mu}{=} Z \vee U, Z \stackrel{\mu}{=} \mathsf{EF}_{\rho_1} Y, U \stackrel{\mu}{=} \mathsf{EF}_{\rho_2} Y\} \rrbracket_K \delta)(X).
  \end{aligned}$$

  By applying the definition of $\llbracket\ \rrbracket$ and simple properties about substitution of variables in a Res, we obtain:

$$
\begin{array}{l}
(\llbracket \{X \stackrel{\mu}{=} \mathsf{EF}_{\rho_1|\rho_2} Y\} \rrbracket_K \delta)(X) \\
= \llbracket \mathsf{EF}_{\rho_1|\rho_2} Y \rrbracket_K \delta \\
= \{s \in S \mid \exists \pi \in Path_K(s). \exists i \geq 0. \pi_{0,i} \models_K \rho_1|\rho_2 \wedge \pi_i \in \llbracket Y \rrbracket_K \delta\} \\
= \{s \in S \mid \exists \pi \in Path_K(s). \exists i \geq 0. (\pi_{0,i} \models_K \rho_1 \vee \pi_{0,i} \models_K \rho_2) \wedge \\
\qquad\qquad \pi_i \in \llbracket Y \rrbracket_K \delta\} \\
= \{s \in S \mid \exists \pi \in Path_K(s). \exists i \geq 0. ((\pi_{0,i} \models_K \rho_1 \wedge \pi_i \in \llbracket Y \rrbracket_K \delta) \vee \\
\qquad\qquad (\pi_{0,i} \models_K \rho_2 \wedge \pi_i \in \llbracket Y \rrbracket_K \delta))\} \\
= \{s \in S \mid \exists \pi \in Path_K(s). \exists i \geq 0. \pi_{0,i} \models_K \rho_1 \wedge \pi_i \in \llbracket Y \rrbracket_K \delta \vee \\
\qquad\qquad \exists \pi \in Path_K(s). \exists i \geq 0. \pi_{0,i} \models_K \rho_2 \wedge \pi_i \in \llbracket Y \rrbracket_K \delta\} \\
= \llbracket \mathsf{EF}_{\rho_1} Y \vee \mathsf{EF}_{\rho_2} Y \rrbracket_K \delta \\
= (\llbracket \{X \stackrel{\mu}{=} \mathsf{EF}_{\rho_1} Y \vee \mathsf{EF}_{\rho_2} Y\} \rrbracket_K \delta)(X) \\
= (\llbracket \{X \stackrel{\mu}{=} Z \vee U, Z \stackrel{\mu}{=} \mathsf{EF}_{\rho_1} Y, U \stackrel{\mu}{=} \mathsf{EF}_{\rho_2} Y\} \rrbracket_K \delta)(X).
\end{array}
$$

- Substitution $X \stackrel{\mu}{=} \mathsf{EF}_{\rho^*} Y := X \stackrel{\mu}{=} Y \vee Z, Z \stackrel{\mu}{=} \mathsf{EF}_{\rho} X$. As above, it is sufficient to show that:

$$
(\llbracket \{X \stackrel{\mu}{=} \mathsf{EF}_{\rho^*} Y\} \rrbracket_K \delta)(X) = (\llbracket \{X \stackrel{\mu}{=} Y \vee Z, Z \stackrel{\mu}{=} \mathsf{EF}_{\rho} X\} \rrbracket_K \delta)(X).
$$

Let $A = (\llbracket \{X \stackrel{\mu}{=} \mathsf{EF}_{\rho^*} Y\} \rrbracket_K \delta)(X)$. We have:

$$
\begin{array}{l}
(\llbracket \{X \stackrel{\mu}{=} \mathsf{EF}_{\rho^*} Y\} \rrbracket_K \delta)(X) = \qquad \text{by def. of } \llbracket\ \rrbracket \\
\llbracket \mathsf{EF}_{\rho^*} Y \rrbracket_K \delta = \\
\{s \in S \mid \exists \pi \in Path_K(s). \exists i \geq 0. \pi_{0,i} \models_K \rho^* \wedge \pi_i \in \llbracket Y \rrbracket_K \delta\} = \\
\{s \in S \mid \exists \pi \in Path_K(s). \exists i \geq 0. \exists k \geq 0. \pi_{0,i} \models_K \rho^k \wedge \pi_i \in \delta(Y)\}.
\end{array}
$$

Let $B = (\llbracket \{X \stackrel{\mu}{=} Y \vee Z, Z \stackrel{\mu}{=} \mathsf{EF}_{\rho} X\} \rrbracket_K \delta)(X)$. We have:

$$
\begin{array}{l}
(\llbracket \{X \stackrel{\mu}{=} Y \vee Z, Z \stackrel{\mu}{=} \mathsf{EF}_{\rho} X\} \rrbracket_K \delta)(X) = \quad \text{by subst. on } Y \\
(\llbracket \{X \stackrel{\mu}{=} Y \vee \mathsf{EF}_{\rho} X\} \rrbracket_K \delta)(X) = \mu \Phi_\delta
\end{array}
$$

where the functional $\Phi_\delta : 2^S \to 2^S$ is defined as follows:

$$
\begin{array}{rl}
\Phi_\delta(U) & = \llbracket Y \vee \mathsf{EF}_{\rho} X \rrbracket_K (\delta \oslash [U/X]) \\
& = \llbracket Y \rrbracket_K (\delta \oslash [U/X]) \cup \llbracket \mathsf{EF}_{\rho} X \rrbracket_K (\delta \oslash [U/X]) \\
& = \delta(Y) \cup \llbracket \mathsf{EF}_{\rho} X \rrbracket_K [U/X].
\end{array}
$$

The lattice $\langle 2^S, \emptyset, S, \cap, \cup \rangle$ being finite, the minimal fixed point $\mu \Phi_\delta$ has also the following iterative characterization [58]:

$$
\mu \Phi_\delta = \bigcup_{k \geq 0} \Phi^k(\emptyset),
$$

where $\Phi^0(\emptyset) = \emptyset$, $\Phi^{k+1}(\emptyset) = \delta(Y) \cup [\![\mathsf{EF}_\rho X]\!]_K[\Phi^k(\emptyset)/X]$.

Intuitively, $\Phi^{k+1}(\emptyset)$ contains those states having an outgoing sequence that matches ρ^j for some $j \in [0,k]$ and leads to a state in $\delta(Y)$:

$$\begin{aligned}\Phi^{k+1}(\emptyset) = \{s \in S \mid \exists \pi \in Path_K(s).\exists i \geq 0.\exists j \in [0,k].\\ \pi_{0,i} \models_K \rho^j \wedge \pi_i \in \delta(Y)\}.\end{aligned}$$

This statement can be easily shown by induction on k.
Base step.

$$\begin{aligned}\Phi^1(\emptyset) &= \delta(Y) \cup [\![\mathsf{EF}_\rho X]\!]_K[\Phi^0(\emptyset)/X]\\ &= \delta(Y) \cup [\![\mathsf{EF}_\rho X]\!]_K[\emptyset/X]\\ &= \delta(Y)\\ &= \{s \in S \mid \exists \pi \in Path_K(s).\pi_{0,0} \models_K \rho^0 \wedge \pi_0 \in \delta(Y)\}\\ &\qquad \text{by choosing } i, j = 0.\end{aligned}$$

Inductive step.

$$\begin{aligned}\Phi^{k+2}(\emptyset) &= \Phi(\Phi^{k+1}(\emptyset)) \qquad \text{by def. of } \Phi\\ &= \delta(Y) \cup \{s \in S \mid \exists \pi \in Path_K(s).\exists i \geq 0.\\ &\qquad\qquad \pi_{0,i} \models_K \rho \wedge \pi_i \in \Phi^{k+1}(\emptyset)\} \qquad \text{by ind. hyp.}\\ &= \delta(Y) \cup \{s \in S \mid \exists \pi \in Path_K(s).\exists i \geq 0.\pi_{0,i} \models_K \rho \wedge\\ &\qquad\qquad \exists \pi' \in Path_K(\pi_i).\exists i' \geq 0.\exists j \in [0,k].\\ &\qquad\qquad \pi'_{0,i'} \models_K \rho^j \wedge \pi'_{i'} \in \delta(Y)\}\\ &= \delta(Y) \cup \{s \in S \mid \exists \pi \in Path_K(s).\exists i \geq 0.\exists j \in [0,k].\\ &\qquad\qquad \pi_{0,i} \models_K \rho^{j+1} \wedge \pi_i \in \delta(Y)\} \quad \text{repl. } i \text{ by } i+i'\\ &= \delta(Y) \cup \{s \in S \mid \exists \pi \in Path_K(s).\exists i \geq 0.\exists j \in [1,k+1].\\ &\qquad\qquad \pi_{0,i} \models_K \rho^j \wedge \pi_i \in \delta(Y)\}\\ &= \{s \in S \mid \exists \pi \in Path_K(s).\exists i \geq 0.\exists j \in [0,k+1].\\ &\qquad\qquad \pi_{0,i} \models_K \rho^j \wedge \pi_i \in \delta(Y)\}.\end{aligned}$$

From the above statement, we obtain:

$$\begin{aligned}B &= \textstyle\bigcup_{k\geq 0} \Phi^k(\emptyset) = \Phi^0(\emptyset) \cup \bigcup_{k\geq 0} \Phi^{k+1}(\emptyset) = \bigcup_{k\geq 0} \Phi^{k+1}(\emptyset)\\ &= \textstyle\bigcup_{k\geq 0}\{s \in S \mid \exists \pi \in Path_K(s).\exists i \geq 0.\exists j \in [0,k].\\ &\qquad \pi_{0,i} \models_K \rho^j \wedge \pi_i \in \delta(Y)\}\\ &= \{s \in S \mid \exists k \geq 0.\exists \pi \in Path_K(s).\exists i \geq 0.\exists j \in [0,k].\\ &\qquad \pi_{0,i} \models_K \rho^j \wedge \pi_i \in \delta(Y)\}\\ &= \{s \in S \mid \exists \pi \in Path_K(s).\exists i \geq 0.\exists k \geq 0.\exists j \in [0,k].\\ &\qquad \pi_{0,i} \models_K \rho^j \wedge \pi_i \in \delta(Y)\}\\ &= \{s \in S \mid \exists \pi \in Path_K(s).\exists i \geq 0.\exists k \geq 0.\pi_{0,i} \models_K \rho^k \wedge \pi_i \in \delta(Y)\}\\ &\qquad \text{choose } j = k\\ &= A.\end{aligned}$$

□

A.2.2. Operators AF_ρ *and* EG_ρ
Translation to guarded form.

Proof *(Lemma 1).* Let K be a Kripke structure, $B = \{X_i \stackrel{\mu}{=} \varphi_i\}_{1\leq i\leq n}$ be an equation block and δ a propositional environment as stated in the hypothesis. It is sufficient to show that the absorption substitution satisfies the condition in the hypothesis of Lemma 5:

$$([\![\{X \stackrel{\mu}{=} X \vee \varphi\}]\!]_K\delta)(X) = ([\![\{X \stackrel{\mu}{=} \varphi\}]\!]_K\delta)(X)$$

which amounts to show, applying the definition of $[\![\ ]\!]$, that:

$$\mu\Phi_\delta = [\![\varphi]\!]_K\delta$$

where the functional $\Phi_\delta : 2^S \to 2^S$ is defined as $\Phi_\delta(U) = [\![X \vee \varphi]\!]_K(\delta \oslash [U/X]) = U \cup [\![\varphi]\!]_K\delta$. The lattice $\langle 2^S, \emptyset, S, \cap, \cup\rangle$ being finite, the minimal fixed point $\mu\Phi_\delta$ has also the following iterative characterization [58]:

$$\mu\Phi_\delta = \bigcup_{k\geq 0} \Phi^k(\emptyset), \quad \text{where} \quad \Phi^0_\delta(\emptyset) = \emptyset, \quad \Phi^{k+1}_\delta(\emptyset) = \Phi^k_\delta(\emptyset) \cup [\![\varphi]\!]_K\delta.$$

To obtain the desired equality, it is therefore sufficient to show that $\Phi^{k+1}_\delta(\emptyset) = [\![\varphi]\!]_K\delta$ for every $k \geq 0$. We proceed by induction on k.

Base step: $\Phi^1_\delta(\emptyset) = \Phi^0_\delta(\emptyset) \cup [\![\varphi]\!]_K\delta = [\![\varphi]\!]_K\delta$.

Inductive step:

$$\begin{aligned} \Phi^{k+1}_\delta(\emptyset) &= \Phi^k_\delta(\emptyset) \cup [\![\varphi]\!]_K\delta && \text{by def.} \\ &= [\![\varphi]\!]_K\delta \cup [\![\varphi]\!]_K\delta && \text{by ind. hyp.} \\ &= [\![\varphi]\!]_K\delta. \end{aligned}$$

□

Determinization.

Several definitions and lemmas are needed in order to prove Proposition 3. Consider a Kripke structure K and the following potentiality RES:

$$\left\{ X_i \stackrel{\mu}{=} \bigvee_{j=1}^{n} (h_{ij} \wedge \mathsf{EF}_{\rho_{ij}} X_j) \vee (h_i \wedge Y) \right\}_{1\leq i\leq n} \qquad (*)$$

where $h_{ij}, h_i \in \mathbf{Bool}$ and ρ_{ij} are regular formulas for all $1 \leq i, j \leq n$. Unguarded occurrences of variables X_j in the right-hand sides of the equations are obtained by taking $\rho_{ij} = \mathsf{nil}$. RESs of the form $(*)$ are encountered throughout the translation from a potentiality RES to its guarded form. For instance, a potentiality RES $\{X_1 \stackrel{\mu}{=} \mathsf{EF}_\rho Y\}$ can be rewritten as $\{X_1 \stackrel{\mu}{=} \mathsf{EF}_\rho X_2, X_2 \stackrel{\mu}{=} Y\}$, which is in the form above by considering $n = 2$, $h_{11} = \mathsf{false}$, $h_{12} = \mathsf{true}$, $\rho_{12} = \rho$, $h_1 = \mathsf{false}$, $h_{21} = h_{22} = \mathsf{false}$, and $h_2 = \mathsf{true}$. Similarly, a guarded potentiality RES is a particular case of form $(*)$ in which all regular formulas ρ_{ij} are simply atomic propositions p_{ij}.

To each propositional variable X_i of the potentiality RES $(*)$ and environment δ is associated a *path predicate* $P_{\delta,i} : Path_K \to \mathbf{Bool}$ characterizing the paths denoted by X_i in the context of δ. These path predicates are defined by the following equation system:

$$\left\{ P_{\delta,i}(\pi) \stackrel{\mu}{=} \bigvee_{j=1}^{n} \left(h_{ij} \wedge \exists l_{ij} \geq 0 . \pi_{0,l_{ij}} \models \rho_{ij} \wedge P_{\delta,j}(\pi_{l_{ij},\infty}) \right) \vee (h_i \wedge \pi_0 \in \delta(Y)) \right\}_{1 \leq i \leq n}$$

where $\pi_{l_{ij},\infty}$ denotes the suffix of path π starting at the state of index l_{ij}.

The translation from a potentiality MES $\{X_1 \stackrel{\mu}{=} \mathsf{EF}_\rho Y\}$ to its guarded form preserves the path predicate associated to the main variable X_1, as shown by the lemma below.

Lemma 6. *Let K be a Kripke structure, $R = \{X_1 \stackrel{\mu}{=} \mathsf{EF}_\rho Y\}$ be an equation block, M be its corresponding guarded potentiality* MES *in the form $(*)$, and $P_{\delta,i}$ be its associated path predicates. Then:*

$$P_{\delta,1}(\pi) = \exists l \geq 0 . \pi_{0,l} \models \rho \wedge \pi_l \in \delta(Y)$$

for any $\pi \in Path_K$ and any propositional environment δ.

Proof Let K be a Kripke structure and δ be a propositional environment. Let the equation block $R = \{X_1 \stackrel{\mu}{=} \mathsf{EF}_\rho X_2, X_2 \stackrel{\mu}{=} Y\}$ in form $(*)$. Its associated path predicates are defined as follows:

$$\begin{aligned} P_{\delta,1}(\pi) &= \exists l_{12} \geq 0 . \pi_{0,l_{12}} \models \rho_{12} \wedge P_{\delta,2}(\pi_{l_{12},\infty}) \\ P_{\delta,2}(\pi) &= \pi_0 \in \delta(Y) \end{aligned}$$

where $\pi \in Path_K$. After appropriate renamings, we obtain the desired equality:

$$P_{\delta,1}(\pi) = \exists l \geq 0 . \pi_{0,l} \models \rho \wedge \pi_l \in \delta(Y).$$

It remains to show that this equality also holds along the translation of R into guarded form. This translation consists of two phases: elimination of the regular operators present in ρ (Proposition 2) and elimination of unguarded occurrences of variables (Lemma 1). The substitutions performed in both phases preserve the path predicates associated to the variables defined by the substituted equations. This can be shown using similar arguments as in Proposition 2 and Lemma 1; we show below the path predicate preservation only for the first rule in Proposition 2, leaving the other ones as exercises for the interested reader.

This rule transforms the RES $R = \{X \stackrel{\mu}{=} \mathsf{EF}_{\rho_1.\rho_2} Y\}$ into the RES $R' = \{X \stackrel{\mu}{=} \mathsf{EF}_{\rho_1} Z, Z \stackrel{\mu}{=} \mathsf{EF}_{\rho_2} Y\}$. The predicate $P'_{\delta,1}$ associated to X_1 in R' is defined as follows:

$$\begin{aligned}
P'_{\delta,1}(\pi) &= \exists l \geq 0.\pi_{0,l} \models \rho_1 \wedge P'_{\delta,2}(\pi_{l,\infty}) && \text{by def. of } P'_{\delta,1} \\
&= \exists l \geq 0.\pi_{0,l} \models \rho_1 \wedge \exists l' \geq 0.\pi_{l,l+l'} \models \rho_2 \wedge \pi_{l+l'} \in \delta(Y) && \text{by def. of } P'_{\delta,2} \\
&= \exists l \geq 0.\exists l' \geq 0.\pi_{0,l} \models \rho_1 \wedge \pi_{l,l+l'} \models \rho_2 \wedge \pi_{l+l'} \in \delta(Y) \\
&= \exists k \geq 0.\exists j \geq 0.\pi_{0,j} \models \rho_1 \wedge \pi_{j,k} \models \rho_2 \wedge \pi_k \in \delta(Y) \\
&\qquad \text{by taking } k = l + l' \text{ and } j = l \\
&= \exists k \geq 0.\exists j \geq 0.\pi_{0,j} \models \rho_1 \wedge \pi_{j,k} \models \rho_2 \wedge \pi_k \in \delta(Y) && \text{by def. of } \rho_1.\rho_2
\end{aligned}$$

which coincides with the definition of $P_{\delta,1}$ in R. Thus, the path predicate $P_{\delta,1}$ associated to X_1 in R remains unchanged during the translation of R into guarded form, which shows the desired equality. □

The relation between the path predicates associated to a guarded potentiality MES and the interpretation of the corresponding determinized MES is given by the lemma below.

Lemma 7. *Let K be a Kripke structure, M be a guarded potentiality* MES *in the form $(*)$, and $P_{\delta,i}$ be its associated path predicates. The determinized* MES *corresponding to M is defined as in Section 3.3.2. Then:*

$$\begin{aligned}
&\Big(\Big[\!\Big[\Big\{X_I \stackrel{\mu}{=} \bigvee_{\emptyset \subset Q \subseteq prop(I)} \mathsf{AF}_Q X_{vars(Q,I)} \vee (h(I) \wedge Y)\Big\}_{I \subseteq [1,n]}\Big]\!\Big]_K \delta\Big)(X_J) \\
&= \\
&\{s \in S \mid \forall \pi \in Path_K(s).\exists j \in J.P_{\delta,j}(\pi)\}
\end{aligned}$$

for any index set $J \subseteq [1,n]$ and any propositional environment δ.

Proof Let K, M, δ, and $P_{\delta,i}$ as stated in the hypothesis. The functional $\Phi_\delta : (2^S)^{2^n-1} \to (2^S)^{2^n-1}$ associated to the determinized MES corresponding

to M is defined as follows:

$$\Phi_\delta(\langle U_J\rangle_{J\subseteq[1,n]}) = \Big\langle \Big[\!\Big[\bigvee_{\emptyset\subset Q\subseteq prop(I)} \mathsf{AF}_Q X_{vars(Q,I)} \vee (h(I)\wedge Y)\Big]\!\Big]_K (\delta \oslash [U_J/X_J]_{J\subseteq[1,n]})\Big\rangle_{I\subseteq[1,n]}$$

The interpretation of the determinized MES is equal to $\mu\Phi_\delta$. Let $U = \langle\{s \in S \mid \forall\pi \in Path_K(s).\exists j \in J.P_{\delta,j}(\pi)\}\rangle_{J\subseteq[1,n]}$. We must show that $\mu\Phi_\delta = U$, which we split into a double inclusion.

Inclusion $\subseteq$.. By Tarski's theorem [78], showing that $\mu\Phi_\delta \subseteq U$ amounts to show that $\Phi_\delta(U) \subseteq U$. We have:

$$\Phi_\delta(U) = \Big\langle \Big[\!\Big[\bigvee_{\emptyset\subset Q\subseteq prop(I)} \mathsf{AF}_Q X_{vars(Q,I)} \vee (h(I)\wedge Y)\Big]\!\Big]_K (\delta \oslash [\{s \in S \mid \forall\pi \in Path_K(s).\exists j \in J.P_{\delta,j}(\pi)\}/X_J]_{J\subseteq[1,n]})\Big\rangle_{I\subseteq[1,n]}$$

Let $I \subseteq [1,n]$ and $s \in (\Phi_\delta(U))_I$. By using the definition of Φ_δ and the interpretation of AF, and doing a simple first order reasoning, this is equivalent to the disjunction of the two conditions below:

(a) $\exists\emptyset \subset Q \subseteq prop(I).(s \models Q \wedge \forall\pi \in Path_K(s).\exists j \in vars(Q,I).P_{\delta,j}(\pi_{1,\infty}))$
(b) $h(I) \wedge s \in \delta(Y)$.

We must show that $s \in U_I$, i.e., that $\forall\pi \in Path(s).\exists i \in I.P_{\delta,i}(\pi)$. By applying the definition of path predicates, this expands as follows:

$$\forall\pi \in Path(s).\exists i \in I.(\exists j \in [1,n].(h_{ij} \wedge s \models p_{ij} \wedge P_{\delta,j}(\pi_{1,\infty})) \vee (h_i \wedge s \in \delta(Y)))$$

which is equivalent to the disjunction of the two conditions below:

(a') $\forall\pi \in Path(s).\exists j \in [1,n].(\exists i \in I.(h_{ij} \wedge s \models p_{ij}) \wedge P_{\delta,j}(\pi_{1,\infty}))$
(b') $\exists i \in I.h_i \wedge s \in \delta(Y)$.

Two cases are possible, depending on the fact that (a) or (b) holds.

Case (a). Let $Q \subseteq prop(I)$ such that $s \in Q$ and for all $\pi \in Path_K(s)$ there exists $j \in vars(Q,I)$ such that $P_{\delta,j}(\pi_{1,\infty})$. Let $\pi \in Path_K(s)$. From condition (a), we can choose $j \in vars(Q,I)$ such that $P_{\delta,j}(\pi_{1,\infty})$. Based on the definition of $vars(Q,I)$, we can choose $i \in I$ such that $p_{ij} \in Q$ and $h_{ij} = \mathsf{true}$. Since $s \models Q$ and $p_{ij} \in Q$, it follows that $s \models p_{ij}$ (recall from Section 3.3.2 that Q stands for the conjunction of all atomic propositions that it contains). This implies condition (a').

Case (b). Assume that $h(I) = \mathsf{true}$ and $s \in \delta(Y)$. From the definition of $h(I)$, it follows that we can choose $i \in I$ such that $h_i = \mathsf{true}$. This implies condition (b').

Inclusion $\supseteq$*..* The equation system defining the path predicates associated to M is defined as follows:

$$\left\{ P_{\delta,j}(\pi) \stackrel{\mu}{=} \bigvee_{k=1}^{n} (h_{jk} \wedge \pi_0 \models p_{jk} \wedge P_{\delta,k}(\pi_{1,\infty})) \vee (h_j \wedge \pi_0 \in \delta(Y)) \right\}_{1 \leq j \leq n}$$

For simplicity, we assume that all predicates occurring in the right-hand sides of equations are defined by some other equations of the system; this corresponds to the fact that M does not have free propositional variables excepting Y, whose interpretation is given by the environment δ. The functional $\Pi_\delta : (Path_K \to \mathbf{Bool})^n \to (Path_K \to \mathbf{Bool})^n$ associated to this system is defined below:

$$\Pi_\delta(P_1, ..., P_n) = \left\langle \lambda\pi. \left(\bigvee_{k=1}^{n} (h_{jk} \wedge \pi_0 \models p_{jk} \wedge P_k(\pi_{1,\infty})) \vee (h_j \wedge \pi_0 \in \delta(Y)) \right) \right\rangle_{1 \leq j \leq n}$$

It is straightforward to check that the functional Π_δ is continuous on the lattice $\langle (Path_K \to \mathbf{Bool})^n, (\lambda\pi.\mathsf{false})^n, (\lambda\pi.\mathsf{true})^n, \sqcap, \sqcup \rangle$, where $\sqcup$ and $\sqcap$ are the pointwise extensions of disjunction and conjunction operations on path predicates. Therefore, its minimal fixed point $\mu\Pi_\delta$, which gives the interpretation of the equation system, has the following iterative characterization [58]:

$$\mu\Pi_\delta = \bigsqcup_{k \geq 0} \Pi_\delta^k((\lambda\pi.\mathsf{false})^n), \qquad \Pi_\delta^0((\lambda\pi.\mathsf{false})^n) = (\lambda\pi.\mathsf{false})^n.$$

We note $\langle P_{\delta,j}^k \rangle_{1 \leq j \leq n} = \Pi_\delta^k((\lambda\pi.\mathsf{false})^n)$. From the iterative characterization of $\mu\Pi_\delta$ and the definition of $\sqcup$, we have:

$$P_{\delta,j}(\pi) = \left(\bigsqcup_{k \geq 0} \langle P_{\delta,j}^k \rangle_{1 \leq j \leq n} \right)(\pi) = \exists k \geq 0. P_{\delta,j}^k(\pi).$$

To obtain the desired inclusion, we use the following statement:

$$\forall k \geq 0. \left(\left\langle \{ s \in S \mid \forall \pi \in Path_K(s). \exists j \in J. P_{\delta,j}^k(\pi) \} \right\rangle_{J \subseteq [1,n]} \subseteq \mu\Phi_\delta \right) \qquad (**)$$

To show that $U \subseteq \mu\Phi_\delta$, let $J \subseteq [1,n]$ and let $s \in U_J$, which means that for all $\pi \in Path_K(s)$, there exists $j \in J$ such that $P_{\delta,j}(\pi)$. The definition of $P_{\delta,j}(\pi)$ above ensures that we can find $k \geq 0$ such that $P^k_{\delta,j}(\pi)$. By applying $(**)$ for that k, we obtain $s \in (\mu\Phi_\delta)_J$, which implies in turn the desired inclusion $U \subseteq \mu\Phi_\delta$.

It remains to show the $(**)$ statement. We proceed by induction on k.

Base step.

$\big\langle\{s \in S \mid \forall \pi \in Path_K(s).\exists j \in J.P^0_{\delta,j}(\pi)\}\big\rangle_{J\subseteq[1,n]} =$ by def. $\Pi^0_\delta((\lambda\pi.\mathsf{false})^n)$

$\langle\{s \in S \mid \forall \pi \in Path_K(s).\exists j \in J.\mathsf{false}\}\rangle_{J\subseteq[1,n]} = \langle\emptyset\rangle_{J\subseteq[1,n]} \subseteq \mu\Phi_\delta.$

Inductive step. Let $U^k = \langle\{s \in S \mid \forall \pi \in Path_K(s).\exists j \in J.P^k_{\delta,j}(\pi)\}\rangle_{J\subseteq[1,n]}$. We show below that $U^{k+1} \subseteq \Phi_\delta(U^k)$, which together with the inductive hypothesis and the definition of minimal fixed points implies $U^{k+1} \subseteq \Phi_\delta(U^k) \subseteq \Phi_\delta(\mu\Phi_\delta) = \mu\Phi_\delta$, i.e., the desired inequality.

Let $J \subseteq [1,n]$ and let $s \in (U^{k+1})_J$, which means that for every $\pi \in Path_K(s)$ there exists $j \in J$ such that $P^{k+1}_{\delta,j}(\pi)$. From the definition of Π_δ and $P^k_{\delta,j}$, we have:

$$P^{k+1}_{\delta,j}(\pi) = \bigvee_{l=1}^{n}(h_{jl} \wedge \pi_0 \models p_{jl} \wedge P^k_{\delta,l}(\pi_{1,\infty})) \vee (h_j \wedge \pi_0 \in \delta(Y)).$$

By expanding this equality and doing a simple first order reasoning, the conditions above can be rewritten as the disjunction of the two conditions below:

(c) $\forall \pi \in Path_K(s).\exists l \in [1,n].(\exists j \in J.(h_{jl} \wedge s \models p_{jl}) \wedge P^k_{\delta,l}(\pi_{1,\infty}))$
(d) $\exists j \in J.h_j \wedge s \in \delta(Y)$.

Let $s \in (\Phi_\delta(U^k))_J$. From the definition of Φ_δ, this is equivalent to:

$$s \in \Big[\!\Big[\bigvee_{\emptyset\subset Q\subseteq prop(J)} \mathsf{AF}_Q X_{vars(Q,J)} \vee (h(J) \wedge Y)\Big]\!\Big]_K (\delta \oslash [(U^k)_L/X_L]_{L\subseteq[1,n]})$$

Using the definition of U^k and the interpretation of AF, and doing a simple first order reasoning, this is equivalent to the disjunction of the two conditions below:

(c') $\exists \emptyset \subset Q \subseteq prop(J).(s \models Q \wedge \forall \pi \in Path_K(s).\exists l \in vars(Q,J).P^k_{\delta,l}(\pi_{1,\infty}))$
(d') $h(J) \wedge s \in \delta(Y)$.

Two cases are possible, depending on the fact that (c) or (d) holds.

Case (c). Let the set of atomic propositions Q be defined as follows:

$$Q = \bigcup_{\pi \in Path_K(s)} \{p_{jl} \mid j \in J \wedge l \in [1,n] \wedge s \models p_{jl}\}$$

Condition (c) guarantees that Q is not empty and the definition of $prop(J)$ implies that $Q \subseteq prop(J)$. Since $s \models p_{jl}$ for every $p_{jl} \in Q$, it follows that $s \models Q$ (recall from Section 3.3.2 that Q stands for the conjunction of all atomic propositions that it contains). Let $\pi \in Path_K(s)$. From condition (c), we can find $l \in [1,n]$ and $j \in J$ such that h_{jl} and $s \models p_{jl}$ and $P_{\delta,j}(\pi_{1,\infty})$. Since $p_{jl} \in Q$ by definition of Q, from the definition of $vars(Q,J)$ it follows that $l \in vars(Q,J)$. This implies condition (c').

Case (d). Let $j \in J$ such that $h_j = \mathsf{true}$. From the definition of $h(J)$, it follows that $h(J) = \mathsf{true}$. Since $s \in \delta(Y)$ from condition (d), this implies condition (d').

This concludes the proof of the lemma. □

We are finally ready to prove Proposition 3.

Proof *(Proposition 3).*

Let K be a Kripke structure, δ be a propositional environment, $R = \{X_1 \stackrel{\mu}{=} \mathsf{AF}_\rho Y\}$ an equation block. Let $P_{\delta,i}$ be the path predicates associated to the guarded potentiality MES obtained by translating R, and let M be the MES further obtained after determinization.

We have:

$([\![M]\!]_K\delta)(X_{\{1\}}) =$	by Lemma 7
$\{s \in S \mid \forall \pi \in Path_K(s).P_{\delta,1}(\pi)\} =$	by Lemma 6
$\{s \in S \mid \forall \pi \in Path_K(s).\exists l \geq 0.\pi_{0,l} \models \rho \wedge \pi_l \in \delta(Y)\} =$	by def. of $\mathsf{AF}_\rho Y$ and $[\![\ ]\!]$
$([\![\{X_1 \stackrel{\mu}{=} \mathsf{AF}_\rho Y\}]\!]_K\delta)(X_1).$	

□