# Computation Tree Regular Logic for Genetic Regulatory Networks

Radu Mateescu, Pedro Monteiro, Estelle Dumas, Hidde de Jong

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

# *Computation Tree Regular Logic for Genetic Regulatory Networks*

Radu Mateescu — Pedro T. Monteiro —

Estelle Dumas — Hidde de Jong

**N° 6521**

Mai 2008

Thèmes BIO et COM


ISSN 0249-6399 ISRN INRIA/RR--6521--FR+ENG

# Computation Tree Regular Logic for Genetic Regulatory Networks

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

Thèmes BIO et COM — Systèmes biologiques et Systèmes communicants
Projets Ibis et Vasy

Rapport de recherche n° 6521 — Mai 2008 — 53 pages

**Abstract:** Model checking has proven to be a useful analysis technique not only for concurrent systems, but also for the genetic regulatory networks (GRNs) that govern the functioning of living cells. The applications of model checking in systems biology have revealed that temporal logics should be able to capture both branching-time and fairness properties (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 report, 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, inspired from the modalities of Propositional Dynamic Logic (PDL). We also develop a translation of CTRL into Hennessy-Milner Logic with Recursion (HMLR), an equational variant of the modal $\mu$-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, system biology, temporal logic, verification

[*] Radu.Mateescu@inria.fr
[**] Pedro.Monteiro@inria.fr
[***] Estelle.Dumas@inria.fr
[****] Hidde.de-Jong@inria.fr

Centre de recherche INRIA Grenoble – Rhône-Alpes
Inovallée, 655, avenue de l'Europe, Montbonnot, 38334 Saint Ismier Cedex (France)
Téléphone : +33 4 76 61 52 00 — Télécopie +33 4 76 61 52 52

# Logique régulière du temps arborescent pour les réseaux de régulation génique

**Résumé :** La vérification par logique temporelle (*model checking*) s'est révélée comme étant une technique d'analyse utile non seulement pour les systèmes concurrents, mais également pour les réseaux de régulation génique (RRGs) qui gouvernent le fonctionnement des cellules vivantes. Les applications du model checking en biologie systémique ont souligné le fait que les logiques temporelles utilisées devraient capturer à la fois des aspects arborescents et d'équité (nécessaires pour spécifier des propriétés de multistabilité et respectivement d'oscillation), tout en ayant une syntaxe simple et intuitive, facile d'emploi pour les non-spécialistes. Dans ce rapport, nous définissons *Computation Tree Regular Logic* (CTRL), une extension de CTL avec des expressions régulières et des opérateurs d'équité qui vise à répondre à ces critères. CTRL subsume à la fois CTL et LTL, et possède un ensemble réduit d'opérateurs temporels indexés par des expressions régulières, inspirés des modalités de *Propositional Dynamic Logic* (PDL). Nous définissons également une traduction de CTRL vers la logique de Hennessy-Milner avec récursion (HMLR), une variante équationnelle du $\mu$-calcul modal, qui a permis d'obtenir un évaluateur à la volée avec diagnostic pour CTRL en réutilisant directement la technologie de vérification disponible dans la boîte à outils CADP. Nous illustrons l'application de l'évaluateur pour CTRL avec l'exemple du RRG contrôlant la réponse au manque de carbone chez *Escherichia coli*.

**Mots-clés :** réseaux de régulation génique, model checking, biologie systémique, logique temporelle, vérification

# 1 Introduction

Explicit state verification has been mostly applied to the analysis of concurrent systems in engineering. Recently, however, biological regulatory networks have been recognized as special cases of concurrent systems as well, which has opened the way for the application of formal verification technology in the emerging field of systems biology (see [29, 54] for reviews). The networks controlling cellular functions consist of genes, proteins, small molecules, and their mutual interactions. Most of these networks are large and complex, thus defying our capacity to understand how the dynamic behavior of the cell emerges from the structure of interactions. A large number of mathematical formalisms have been proposed to describe these networks [23], giving rise to models that can be directly or indirectly mapped to Kripke structures.

The representation of the dynamics of biological regulatory networks by means of Kripke structures enables the application of formal verification techniques to the analysis of properties of the networks. 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? [43]
- 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? [19].
- Does Shc phosphorylation exhibit a relative acceleration with decreasing EGF concentration and show a decline over time? [56]

Several applications of model checking exists in the bioinformatics and systems biology literature [3, 6, 10, 13, 16, 17, 30]. In our previous work [8, 10], we have developed Genetic Network Analyzer (GNA), a tool for the qualitative simulation of genetic regulatory networks, and connected it to state-of-the-art model checkers like NUSMV [20] and CADP [33].

All of the above approaches express the properties of interest in classical temporal logics like CTL [21] and LTL [48]. 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 [59], 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 [19, 47]. CTL is not expressive enough to specify the occurrence of oscillations of indefinite length, a special kind of fairness property [10]. An obvious solution would be to consider CTL$^*$ [26] or the propositional $\mu$-calculus [44], both of which subsume CTL and LTL; however, these powerful branching-time logics are complex to understand and use by non-experts. More generally, it is not easy

to express observations in temporal logic. Often these take the form of patterns of events corresponding to variations of system parameters (protein concentrations, their derivatives, etc.), which can be compared with the model predictions and thus help validate the model. Observations are conveniently and concisely formulated in terms of regular expressions, but these are not provided by standard temporal logics such as CTL and LTL.

In this report, 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 [12] and RegCTL [14], whilst having a simpler syntax due to a different choice of primitive temporal operators, inspired from dynamic logics like PDL [28]. CTRL also subsumes CTL, LTL, and PDL-$\Delta$ [57] allowing in particular the concise expression of bistability and oscillation properties. Although CTRL was primarily designed for describing properties of regulatory networks in system biology, it also enables a succinct formulation of typical safety, liveness, and fairness properties useful for the verification of concurrent systems in other domains.

As regards the evaluation of CTRL formulas on Kripke structures, we 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 [33], 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 $\mu$-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 ($\mu$-calculus and LTSs). The translation from Kripke structures to LTSs is done in the standard way [21], 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 [46], an alternative equational representation of the modal $\mu$-calculus which is also accepted as input by the EVALUATOR 3.6 [51] model checker of CADP. HMLR yields a more succinct translation than plain $\mu$-calculus due to the possibility of factoring out common subformulas using fixed point equations. Moreover, the equational representation of HMLR is closer to the boolean equation systems (BESs) used as intermediate formalism by the verification engine, namely the CÆSAR_SOLVE [50] generic library for local BES resolution. Once the translator from CTRL to HMLR is available, its connection with EVALUATOR 3.6 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-$\Delta$ and allows the multistability and oscillation properties to be captured. The inevitability operator of CTRL and its infinitary version (inevitable looping) has an exponential worst-case complexity w.r.t. the size of its regular subformula; this complexity becomes linear, however, when the regular subformula is "deterministic" in

a way similar to finite automata. In practice, the usage of CTRL and the model checker reveals that properties of biological interest can be expressed and verified efficiently. We illustrate this on the analysis of a model of the regulatory network (GRN) controlling the carbon starvation response of *E. coli*.

**Report outline.** Section 2 defines the syntax and semantics of CTRL and discusses its expressiveness w.r.t. existing widely-used logics. Section 2.4 defines the regular equation systems (RESs), an intermediate equational form into which CTRL formulas will be translated. Sections 3.1 and 3.2 present the translations from CTRL to RESs and then to modal equation systems (MESs). Section 4 formulates the model checking of a MES on a Kripke structure as the resolution of a boolean equation system (BES). We show how this is performed on-the-fly using specialized algorithms and indicate the complexity of the model checking procedure. Section 5 describes the implementation of the CTRL model checker in connection with CADP and illustrates its application on the example of *E. coli*. Section 6 provides some concluding remarks and directions for future work. Appendix A contains the proofs of the translation phases from CTRL to MESs.

# 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 $\pi$ is noted $\pi_i$. The interval going from the $i$-th state of a path $\pi$ to the $j$-th state of $\pi$ inclusively (where $i \leq j$) is noted $\pi_{i,j}$. An interval $\pi_{0,i}$ is called prefix of $\pi$. For each state $s \in S$, $Path(s)$ denotes the set of all paths going out of $s$, i.e., the paths $\pi$ 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 $\varphi$) and *regular formulas* (noted $\rho$), 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 $\mathsf{EF}$, $\mathsf{AF}$, $\mathsf{EF}^\infty$, $\mathsf{AF}^\infty$ temporal operators indexed by regular formulas $\rho$. 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 $\varphi$. 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 $\rho$ (notation $\pi_{i,j} \models_K \rho$). The notation $\rho^j$ (where $j \geq 0$) stands for the concatenation $\rho \dots \rho$, where $\rho$ occurs $j$ times. The semantics of boolean operators is defined in the standard way. A state satisfies the potentiality formula $\mathsf{EF}_\rho\varphi$ iff it has an outgoing path containing a prefix satisfying $\rho$ and leading to a state satisfying $\varphi$. A state satisfies the inevitability formula $\mathsf{AF}_\rho\varphi$ iff all of its outgoing paths contain a prefix satisfying $\rho$ and lead to a state satisfying $\varphi$. A state satisfies the potential looping formula $\mathsf{EF}^\infty_\rho$ iff it has an outgoing path consisting of an infinite concatenation of intervals satisfying $\rho$. A state satisfies the inevitable looping formula $\mathsf{AF}^\infty_\rho$ iff all of its outgoing paths consist of an infinite concatenation of intervals satisfying $\rho$. An interval satisfies the one-step interval formula $\varphi$ iff it consists of two states, the first of which satisfies $\varphi$. An interval satisfies the concatenation formula $\rho_1.\rho_2$ if it is the concatenation of two subintervals, the first one satisfying $\rho_1$ and the second one satisfying $\rho_2$. An interval satisfies the choice formula $\rho_1|\rho_2$ iff it satisfies either $\rho_1$, or $\rho_2$. An interval satisfies the iteration formula $\rho^*$ iff it is the concatenation of (0 or more) subintervals satisfying $\rho$. By definition, an empty interval $\pi_{i,i}$ satisfies $\rho^0$ for any regular formula $\rho$. A Kripke structure $K$ satisfies a state formula $\varphi$ (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 $\rho$ formulas). They express that for some (resp. each) path going out of a state, all of its prefixes satisfying $\rho$ lead to states satisfying $\varphi$. 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 $\rho^*$ such that no other larger prefix satisfies $\rho^*$; in other words, only a finite number of intervals satisfying $\rho$ can be concatenated at the beginning of the path. The empty interval operator $\mathsf{nil}$ is defined as the iteration (0 or more times) of the $\mathsf{false}$ proposition; an interval satisfies the formula $\mathsf{nil}$ iff it contains a single state. The iteration (1 or more times) operator '+' is defined in the standard way; an interval satisfies $\rho^+$ iff it is the concatenation of (1 or more) intervals satisfying $\rho$.

For technical reasons related to the translation of CTRL formulas to MESs in the model checking process (Section 3), we need to extend the grammar of state formulas with propositional variables $X \in \mathcal{X}$, which denote sets of states:

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

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 in order to take into account the presence of propositional variables: $[\![\varphi]\!]_K\delta$ denotes the set of states satisfying $\varphi$ in the context of $\delta$, which must map every variable occurring in $\varphi$ to a state set. The interpretation of propositional variables is defined as follows:

$$[\![X]\!]_K\delta = \delta(X)$$

| | |
|---|---|
| $\mathsf{true} = p \vee \neg p$ | (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) |
| $\mathsf{EG}^\dashv_\rho = \neg\mathsf{AF}^\infty_\rho$ | (potential saturation) |
| $\mathsf{AG}^\dashv_\rho = \neg\mathsf{EF}^\infty_\rho$ | (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

The interpretation of the other state formulas defined in Figure 1 does not change, except that an extra parameter $\delta$ is added to the interpretation $[\![\ ]\!]$. In the sequel, we assume that all occurrences of propositional variables are positive, i.e., they fall under an even number of negations. This condition (ensured during the translation to MESS) is called *syntactic monotonicity* and was proposed initially to ensure the well-definedness of propositional $\mu$-calculus formulas [44].

For simplicity, we first transform state formulas in *positive normal form* (PNF) by propagating the negations downwards until they reach the atomic formulas (i.e., propositions $p$ or variables $X$) by applying the rules in Figure 2. Given that state formulas are syntactically monotonic, negations can occur after propagation only in front of atomic propositions $p$; for convenience, we also include in the set $P$ the negations of all propositions $p$, as well as the boolean constants $\mathsf{true}$ and $\mathsf{false}$. State formulas in PNF are thus composed of atomic propositions, disjunctions and conjunctions, and all primitive and derived temporal operators of CTRL 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 communication protocols and concurrent systems. Other examples of properties, concerning the behaviour of genetic regulatory networks, can be found in Section 5.2.

**Safety properties.** Informally, these properties specify that "something bad never happens" during the execution of the system. They can be expressed in CTRL by identifying the sequences of states corresponding to safety violations, characterizing them using a regular formula $\rho$, and forbidding their existence in the Kripke structure by checking the formula $\mathsf{AG}_\rho\mathsf{false}$. For example, the CTRL formula below expresses the alternation between emissions

and receptions of messages in a communication protocol that behaves as a one-slot buffer:

$$\mathsf{AG}_{((\mathsf{nil} \mid (\mathsf{true}^*.rcv)).(\neg snd)^*.rcv) \mid (\mathsf{true}^*.snd.(\neg rcv)^*.snd)}\mathsf{false}$$

where the atomic propositions *snd* and *rcv* indicate the emission and reception of a message, respectively. The first alternative in the regular subformula above forbids the occurrence of a reception before the first message was emitted and also the occurrence of two consecutive receptions without an emission in between. The second alternative requires that a reception must occur between every two consecutive emissions.

This property can also be specified in CTL using 5 temporal operators:

$$\neg\mathsf{E}[\neg snd\ \mathsf{U}\ rcv] \wedge \mathsf{AG}(rcv \Rightarrow \neg\mathsf{E}[\neg snd\ \mathsf{U}\ rcv]) \wedge \mathsf{AG}(snd \Rightarrow \neg\mathsf{E}[\neg rcv\ \mathsf{U}\ snd])$$

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 execution of the system. They can be expressed in CTRL by capturing the desirable sequences of states, characterizing them using a regular formula $\rho$, and expressing their potential or inevitable presence in the Kripke structure by means of the $\mathsf{EF}_\rho$ and $\mathsf{AF}_\rho$ operators, respectively. For instance, the CTRL property below states that every time a message is emitted, then it will be eventually received after a finite number of transmission errors:

$$\mathsf{AG}_{\mathsf{true}^*.snd}\mathsf{AF}_{(\mathsf{true}^*.err)^*.rcv}\mathsf{true}$$

This property cannot be specified in CTL because of the presence of two nested $*$ operators in the regular subformula of the AF operator.

**Fairness properties.** Informally, these properties specify the progression of all the concurrent processes present in the system, which may be competing for the access to shared resources. In CTRL, fairness properties can be expressed by identifying the infinite execution sequences denoting the *starvation* of a certain process, characterizing them using the $\mathsf{EF}^\infty_\rho$ operator, and forbidding their presence in the Kripke structure. The CTRL formula below specifies that after a process has requested the access to a shared resource, it cannot be indefinitely preempted in getting access to the resource by another process:

$$\neg\mathsf{EF}_{\mathsf{true}^*.req_1}\mathsf{EF}^\infty_{(\neg get_1)^*.req_2.(\neg get_1)^*.get_2}$$

where the atomic propositions $req_i$ and $get_i$ denote the request and the access of process $i$ to the resource, respectively. An equivalent formulation of this property can be rewritten by propagating the negation through the temporal operators using the dualities shown in Figure 2:

$$\mathsf{AG}_{\mathsf{true}^*.req_1}\mathsf{AG}^\dashv_{(\neg get_1)^*.req_2.(\neg get_1)^*.get_2}$$

This property cannot be expressed in CTL because it involves the repetition of alternating $req_2$ and $get_2$, but can be expressed in LTL using 5 temporal operators:

$$\mathsf{G}(req_1 \Rightarrow ((get_1\ \mathsf{R}\ \neg req_2) \vee (\neg get_1\ \mathsf{U}\ ((req_2 \wedge (get_1\ \mathsf{R}\ \neg get_2)) \vee (get_2 \wedge (get_1\ \mathsf{R}\ \neg req_2))))))$$

where $\varphi_1\ \mathsf{R}\ \varphi_2 = \neg(\neg\varphi_1\ \mathsf{U}\ \neg\varphi_2)$ is the release operator of LTL.

### 2.3 Expressiveness

CTRL is a natural extension of CTL [21], 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 [12] and RegCTL [14], which keep the U operator primitive as in the original logic.

CTRL also subsumes LTL [48], because the potential looping operator $\mathsf{EF}^\infty$ 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 $\mathsf{EF}^\infty$ operator does not allow a direct translation of the LTL operators, but may serve as an intermediate form for model checking LTL formulas; in this respect, this operator is similar to the "never claims" used for specifying properties in the early versions of the SPIN model checker [37].

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) [21]. In fact, the CTRL fragment containing the boolean connectors and the temporal operators EF and $\mathsf{EF}^\infty$ can be seen as a state-based variant of PDL-$\Delta$ [57], which has been shown to be more expressive than CTL$^*$ [62].

As regards other existing extensions of CTL with regular operators, CTRL 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:

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

The subsumption of RegCTL is strict because the U operator of RegCTL cannot describe an infinite concatenation of intervals satisfying a regular formula $\rho$, which is specified in CTRL using the $\mathsf{EF}^\infty_\rho$ operator. In [14] it is shown that RegCTL is more expressive than RCTL [12], the extension of CTL with regular expressions underlying the SUGAR [11] specification language; consequently, CTRL also subsumes RCTL.

### 2.4 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. As intermediate language, we use *regular equation systems* (RESs), which are the propositional counterpart of the PDLR (PDL with recursion) specifications introduced in [51]. 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 $\mu$ and $\nu$ denote minimal and maximal fixed points, respectively. The free and bound variables in a block list are defined as follows:

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

The set $fv(\varphi_i)$ contains all propositional variables occurring in $\varphi_i$. A block list $BL$ is *closed* if $fv(BL) = \emptyset$. In the sequel, 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 contained in $BL$, 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 $BL$.

SYNTAX

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

SEMANTICS

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

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

Figure 3: Syntax and semantics of regular equation systems

The 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 $\delta$ 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 $\delta$ with $[U_1/X_1, ..., U_n/X_n]$, i.e., an environment identical to $\delta$ 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 $\delta$ 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 [58] 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 [46] as an equivalent equational definition of the modal $\mu$-calculus. In our setting, MESs are suitable as target language for translating CTRL formulas, since they are sufficiently close to the boolean equation systems used to represent the model checking problem.

# 3 Translation from CTRL to modal equation systems

The translation of a CTRL state formula $\varphi$ into a MES involves two steps: first the formula is translated into a RES, and then the RES is transformed to 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.

## 3.1 Translation to regular equation systems

The translation of a CTRL state formula $\varphi$ 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 $\varphi$ 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 $\varphi$ does not contain propositional variables, then $bv(t_{BL}(\varphi)) = \emptyset$, i.e., the block list produced by the translation is closed. 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 $\varphi$ a state formula of* CTRL. *Then:*

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

*for any propositional environment $\delta$.*

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

Figure 4: Translation of CTRL formulas into RESs

To illustrate the translation of CTRL formulas into RESs, we consider a branching-time property of biological interest, called *bistability property* [59, 25], which specifies that after an initial state, two different equilibrium states can be potentially reached. This property can be expressed in CTRL by the following formula:

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

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 occurrence of the ';' operator produced by the translation of $\mathsf{EF}_{\mathsf{true}^*}\, eql_1 \wedge \mathsf{EF}_{\mathsf{true}^*}\, eql_2$ 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 $\varphi$, because every rule given in Figure 4 creates, for each operator present in $\varphi$, one block containing a single equation with one operator in its right-hand side.

For simplicity, the translation of a state formula $\varphi$ given in Figure 4 does not take care of the state subformulas $\psi$ that may occur inside the regular formulas $\rho$. However, these subformulas must also be translated into RESs in order to be evaluated on a Kripke structure $K$ during the model checking procedure. This is done by applying the translation recursively on every subformula $\psi$ of a regular formula $\rho$, 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 $\psi$, as illustrated by the formula below:

$$\mathsf{EF}_{(\mathsf{AG}_{\mathsf{true}^*}p)^*}q$$

whose translation yields the following 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 $\psi$ are closed, and hence their main variables can be evaluated independently from the RES $t(\varphi)$. This allows to safely replace each subformula $\psi$ by a "fresh" atomic proposition $p_\psi$, 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.

## 3.2 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 the RES equation blocks into MESs is performed by repeatedly applying various transformations on the equation block, most of them being substitutions of equations.

### 3.2.1 Operators $\mathsf{EF}_\rho$ and $\mathsf{AG}_\rho$

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 $\rho$ by repeatedly applying appropriate substitutions. Each equation containing an $\mathsf{EF}_\rho$ or $\mathsf{AG}_\rho$ 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 |
|---|---|
| $X \stackrel{\mu}{=} \mathsf{EF}_{\rho_1.\rho_2} Y$ | $\{X \stackrel{\mu}{=} \mathsf{EF}_{\rho_1} Z, Z \stackrel{\mu}{=} \mathsf{EF}_{\rho_2} Y\}$ |
| $X \stackrel{\mu}{=} \mathsf{EF}_{\rho_1\|\rho_2} Y$ | $\{X \stackrel{\mu}{=} Z \vee U, Z \stackrel{\mu}{=} \mathsf{EF}_{\rho_1} Y, U \stackrel{\mu}{=} \mathsf{EF}_{\rho_2} Y\}$ |
| $X \stackrel{\mu}{=} \mathsf{EF}_{\rho^*} Y$ | $\{X \stackrel{\mu}{=} Y \vee Z, Z \stackrel{\mu}{=} \mathsf{EF}_{\rho} X\}$ |
| $X \stackrel{\nu}{=} \mathsf{AG}_{\rho_1.\rho_2} Y$ | $\{X \stackrel{\nu}{=} \mathsf{AG}_{\rho_1} Z, Z \stackrel{\nu}{=} \mathsf{AG}_{\rho_2} Y\}$ |
| $X \stackrel{\nu}{=} \mathsf{AG}_{\rho_1\|\rho_2} Y$ | $\{X \stackrel{\nu}{=} Z \wedge U, Z \stackrel{\nu}{=} \mathsf{AG}_{\rho_1} Y, U \stackrel{\nu}{=} \mathsf{AG}_{\rho_2} Y\}$ |
| $X \stackrel{\nu}{=} \mathsf{AG}_{\rho^*} Y$ | $\{X \stackrel{\nu}{=} Y \wedge Z, Z \stackrel{\nu}{=} \mathsf{AG}_{\rho} X\}$ |

Figure 5: Substitutions for the $\mathsf{EF}_\rho$ and $\mathsf{AG}_\rho$ 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 $\delta$, the interpretation of $B_1$ (resp. $B_2$) w.r.t. $\delta$ 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 encoding 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 $\varphi$), since each substitution in Figure 5 replaces a regular operator by at most two variables and two temporal operators $\mathsf{EF}$ (resp. $\mathsf{AG}$).

### 3.2.2 Operators $\mathsf{AF}_\rho$ and $\mathsf{EG}_\rho$

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 $\mathsf{EF}_\rho$ and $\mathsf{AG}_\rho$ counterparts, because the substitutions given in Figure 5 to eliminate the regular expressions $\rho$ are no longer valid for the $\mathsf{AF}_\rho$ and $\mathsf{EG}_\rho$ operators. We consider below only blocks of the form $\{X \stackrel{\mu}{=} \mathsf{AF}_\rho Y\}$, the processing of their $\mathsf{EG}_\rho$ 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 $\rho$ using the substitutions given in Section 3.2.1. 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 $\mathsf{EF}$ operators in the right-hand sides of equations by appropriate occurrences of $\mathsf{AF}$ 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 example of 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 $\rho$ before reaching a state satisfying $Y$, whereas the substitutions in Figure 5 allow to eliminate $\rho$ on individual sequences only. To avoid this difficulty, we switch temporarily to the potentiality form $\{X \stackrel{\mu}{=} \mathsf{EF}_\rho Y\}$, we eliminate $\rho$ by applying the substitutions, and we continue working with the resulting potentiality MES, which characterizes the existence of individual sequences satisfying $\rho$. 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 3.2.1. Figure 6 shows the potentiality MES obtained from the equation block considered above by switching to potentiality form and applying the substitutions in Figure 5 (all equations have the sign $\mu$, which was 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 $\mathsf{EF}$ 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 into guarded form consists in eliminating all unguarded occurrences of variables in the right-hand sides of equations, by applying 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 $\delta$.*

The equations of a potentiality MES have two possible forms, according to the right-hand side formulas produced by the rules in Figure 5: either unguarded (i.e., containing disjunctions of variables in their right-hand side), or guarded (i.e., containing a single occurrence of an $\mathsf{EF}$ operator in their right-hand side). The elimination of unguarded occurrences of variables is carried out by Algorithm 1.

**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**

The first loop of Algorithm 1 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.

| 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 in guarded form ($1^{st}$ part)

| 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 in guarded form ($2^{nd}$ part)

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

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

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

This guarded MES is equivalent to the equation block $\{X \stackrel{\mu}{=} \mathsf{EF}_{(q|p^*)^*.(qr^*).(p^*|q^*)} Y\}$ used by the translation in 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 $\mathsf{EF}$ operator. In this respect, guarded potentiality MESs are similar to the equation systems defining the derivatives of regular expressions [15].

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 on practical examples, we observed 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 $\mathsf{true}$ (and similarly for the unguarded occurrences of $Y$ with their $h_i$ equal to $\mathsf{true}$) 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 $\mathsf{EF}$ 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 $\mathsf{EF}$ 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 $\mathsf{true}$ iff $Y$ occurs unguarded in some equation having its index in the set $I$.

In the $\mathsf{AF}$ 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 $\delta$.*

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

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

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

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

certain atomic propositions are mutually exclusive, which allows to remove the $\mathsf{AF}$ operators whose index subformulas contain those propositions together.

The determinization of a guarded potentiality Mes defined above is similar to the subset construction procedure used for determinizing finite automata [1]. 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 $\rho$. However in practice, the size of determinized Mess obtained after simplifications is close to linear w.r.t. the size of $\rho$, as illustrated by the final Mes shown in Figure 11. When $\rho$ 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 $\rho$.

### 3.2.3 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 $\Phi_e$ and $\Phi_a$, which are defined in turn 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 $\mu$-calculus fragment of alternation depth 2 [27], which allows one level of mutual recursion between minimal and maximal fixed points[1]. The complexity of checking $L\mu_2$ formulas on a Kripke structure $K$ is in general quadratic in the size of $K$; however, we can exploit the particular structure of $R_e$ and $R_a$ in order to devise linear-time on-the-fly model checking algorithms for the $\mathsf{EF}^{\infty}_{\rho}$ operator and (when $\rho$ is deterministic) for the $\mathsf{AF}^{\infty}_{\rho}$ operator, as shown in Section 4.

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 On-the-fly model checking

Given a Kripke structure $K = \langle S, P, L, T, s_0\rangle$ and a Ctrl state formula $\varphi$, the on-the-fly model checking problem consists in determining whether the initial state $s_0$ of $K$ satisfies $\varphi$ by exploring the transition relation $T$ in a forward manner starting at $s_0$. Our objective is to reuse the on-the-fly model checking technology already available in the setting of the Mcl property specification language [52, 51], an extension of L$\mu_1$ with various constructs, among which the regular expressions and fairness operators of Pdl-$\Delta$. Mcl is interpreted on labeled transition systems (Ltss) modeling the behaviour of concurrent programs written in action-based specification languages, such as process algebras. The model checking procedure of Mcl relies upon a translation to the alternation-free fragment of HmlR, which is an equivalent equational representation of L$\mu_1$. The on-the-fly model checking problem of a HmlR specification on an Lts is subsequently rephrased as the local resolution of an alternation-free boolean equation system (Bes), which is carried out using specialized linear-time algorithms [50]. Therefore, in order to reuse this technology it is necessary to switch from Kripke structures to Ltss and to translate Ctrl formulas into HmlR specifications.

Kripke structures are converted into Ltss in the classical way [21], i.e., for each state $s$, the atomic propositions holding at $s$ in the Kripke structure are migrated to the actions labeling the transitions going out of $s$ in the Lts. Formally, the Lts corresponding to a Kripke

[1] The $\mathsf{EF}^{\infty}_{\rho}$ and $\mathsf{AF}^{\infty}_{\rho}$ operators belong strictly to $L\mu_2$ only when $\rho$ contains iteration operators; otherwise, they belong to $L\mu_1$, the $\mu$-calculus fragment of alternation depth 1, which has a linear-time model checking complexity [22].

structure is defined as $\langle S, 2^P, \{(s_1, L(s_1), s_2) \in S \times 2^P \times S \mid s_1 \rightarrow_T s_2\}, s_0\rangle$, where $2^P$ is the set of actions (sets of atomic propositions). This translation is succinct, i.e., every state and transition of the Kripke structure is mapped to a state and a transition of the LTS. It is also suitable for on-the-fly verification because it can be carried out during a forward traversal of the transition relation. As regards the translation of CTRL state formulas to HMLR, we briefly explain below how this is done for each temporal operator.

## 4.1 Operators $\mathsf{EF}_\rho$, $\mathsf{AF}_\rho$, $\mathsf{EG}_\rho$, and $\mathsf{AG}_\rho$

The formulas occurring in the MESs obtained from the $\mathsf{EF}_\rho$, $\mathsf{AF}_\rho$, $\mathsf{EG}_\rho$, and $\mathsf{AG}_\rho$ operators by applying the translation defined in Sections 3.1 and 3.2 contain combinations of atomic propositions and basic modalities of the form $\mathsf{EF}_p X$, $\mathsf{AF}_p X$, $\mathsf{EG}_p X$, and $\mathsf{AG}_p X$, respectively. To convert these MESs into HMLR specifications, we must replace each of these subformulas by HML modalities that have the same interpretation on the LTS corresponding to the original Kripke structure. This is done as indicated in Table 1.

Table 1: Correspondence between CTRL and HML modalities

| CTRL formula | HML formula |
|---|---|
| $p$ | $\langle p\rangle\mathsf{true}$ |
| $\mathsf{EF}_p X$ | $\langle p\rangle X$ |
| $\mathsf{AG}_p X$ | $[p]X$ |
| $\mathsf{AF}_p X$ | $\langle p\rangle\mathsf{true} \wedge [\mathsf{true}]X$ |
| $\mathsf{EG}_p X$ | $\langle p\rangle\mathsf{true} \Rightarrow \langle\mathsf{true}\rangle X$ |

This translation causes at most a linear increase in size of the HMLR specification w.r.t. the MES, each basic CTRL modality being replaced by at most two HML modalities. The resulting HMLR specification is alternation-free, since it has the same equation block structure as the MES. Therefore, the on-the-fly model checking procedure obtained via a BES encoding has a linear-time complexity w.r.t. the size of the HMLR specification (hence, the size of the MES) and the size of the LTS (hence, the size of the Kripke structure) [22, 51].

## 4.2 Operators $\mathsf{EF}^\infty_\rho$, $\mathsf{AF}^\infty_\rho$, $\mathsf{EG}^\dashv_\rho$, and $\mathsf{AG}^\dashv_\rho$

As shown in Section 3.2.3, the looping and saturation operators of CTRL lead to MESs of alternation depth 2, a class having in general a quadratic model checking complexity w.r.t. the size of the Kripke structure. However, given the simple structure of these MESs, we can obtain linear-time on-the-fly model checking procedures by enhancing the BES resolution algorithms already available for solving disjunctive and conjunctive BESs [50], as explained below. For conciseness, we do not present here in full detail the encoding of the model checking problem in terms of BESs (see [22, 51, 50] for details) but instead we illustrate

it on examples. We consider only the looping operators $\mathsf{EF}^{\infty}_{\rho}$ and the $\mathsf{AF}^{\infty}_{\rho}$ operators, the saturation operators $\mathsf{AG}^{\dashv}_{\rho}$ and $\mathsf{EG}^{\dashv}_{\rho}$ being handled dually.

### 4.2.1 Operator $\mathsf{EF}^{\infty}_{\rho}$

The $\mathsf{EF}^{\infty}_{\rho}$ operator yields a RES of the form $\{X \stackrel{\nu}{=} \mathsf{EF}_{\rho} X\}$. To evaluate this RES on a Kripke structure (or equivalently, the corresponding HMLR specification on the LTS derived from the Kripke structure), we abusively switch its sign to $\mu$ and take care to preserve its original meaning during the resolution of the BES encoding the model checking problem. The MES obtained after applying the translation given in Section 3.2.1 contains only disjunctions and $\mathsf{EF}$ operators, which yield diamond modalities in the HMLR specification. Therefore, the BES encoding the model checking problem is disjunctive, and could be solved using the algorithm A4 proposed in [50].

If the $\mathsf{EF}^{\infty}_{\rho}$ formula is false, the solution of the $\{X \stackrel{\mu}{=} \mathsf{EF}_{\rho} X\}$ is also false, since by switching the sign from $\nu$ to $\mu$ we obtain an equation block with a "smaller" interpretation. If $\mathsf{EF}^{\infty}_{\rho}$ is true, the Kripke structure contains an infinite sequence made of subsequences satisfying $\rho$, which must end with a cycle because the set of states is finite. The model checking algorithm must therefore detect the presence of such cycles and record that all the states occurring on them satisfy $\mathsf{EF}^{\infty}_{\rho}$. This cycle detection can be performed by the $\mathrm{A4}_{cyc}$ algorithm proposed in [52], which is an extended version of the A4 algorithm dedicated to disjunctive BESs.

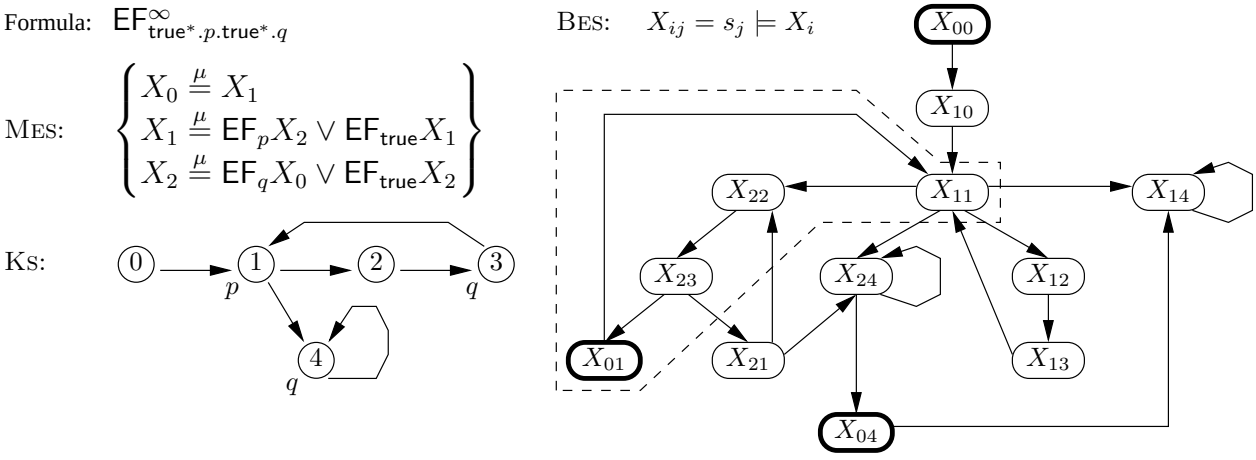


Figure 12: Evaluation of a $\mathsf{EF}^{\infty}$ formula using the $\mathrm{A4}_{cyc}$ algorithm

Figure 12 illustrates the execution of $\mathrm{A4}_{cyc}$ for checking a $\mathsf{EF}^{\infty}_{\rho}$ formula on a Kripke structure. For simplicity, we show the verification by considering directly the RES and the Kripke structure instead of the corresponding HMLR specification and LTS. The verification problem is reformulated as the resolution of a BES in which a variable $X_{ij}$ is true iff the state $s_i$ of the Kripke structure satisfies the propositional variable $X_j$ of the RES. The BES is represented

by means of its associated *boolean graph* [2], which gives a more intuitive representation of the dependencies between boolean variables. The variable $X_0$ of the RES was defined initially by the equation $X_0 \stackrel{\mu}{=} \mathsf{EF}_\rho X_0$ and therefore it plays a special role: every sequence of dependencies in the boolean graph relating two variables $X_{0j}$ denotes a sequence of transitions matching $\rho$ in the Kripke structure. Algorithm A4$_{cyc}$ marks all variables $X_{0j}$ and searches for cycles containing a marked variable: if such a cycle exists, then an infinite sequence satisfying $\rho$ exists in the Kripke structure and the formula is true. In the example shown on Figure 12, the boolean graph associated to the disjunctive BES contains a cycle (surrounded by the dashed line) going through the marked variable $X_{01}$, meaning that the property holds on the Kripke structure.

The A4$_{cyc}$ algorithm has a linear time complexity w.r.t. the size of the boolean graph [52], which makes the complexity of evaluating an $\mathsf{EF}^\infty_\rho$ formula linear w.r.t. the size of $\rho$ and of the Kripke structure.

### 4.2.2 Operator $\mathsf{AF}^\infty_\rho$

**Deterministic case.** The $\mathsf{AF}^\infty_\rho$ operator yields a RES of the form $\{X \stackrel{\nu}{=} \mathsf{AF}_\rho X\}$. To evaluate this RES on a Kripke structure (or equivalently, the corresponding HMLR specification on the LTS derived from the Kripke structure), we abusively keep its sign to $\nu$ and take care to preserve its original meaning during the resolution of the BES encoding the model checking problem. The MES obtained after applying the translation given in Section 3.2.2 (which should have a sign $\mu$) contains disjunctions of $\mathsf{AF}$ operators, each of them yielding box modalities in the HMLR specification. When $\rho$ is deterministic, only one of the $\mathsf{AF}$ operators in a right-hand side of an equation will apply on the current state, leading to a conjunctive BES, which could be solved using the algorithm A4 proposed in [50].

If the $\mathsf{AF}^\infty_\rho$ formula is true, the solution of the $\{X \stackrel{\nu}{=} \mathsf{AF}_\rho X\}$ is also false, since by keeping the sign equal to $\nu$ we obtain an equation block with a "larger" interpretation. If $\mathsf{AF}^\infty_\rho$ is false, the Kripke structure contains a cycle in which one of the $*$-operators of $\rho$ is "trapped", preventing the system to evolve eventually towards the end of the sequences satisfying $\rho$. The model checking algorithm must therefore detect the presence of such cycles and record that all the states occurring on them do not satisfy $\mathsf{AF}^\infty_\rho$. This cycle detection can be performed by a symmetric version of the A4$_{cyc}$ algorithm proposed in [52].

Figure 13 illustrates the execution of the symmetric version of A4$_{cyc}$ for checking a $\mathsf{AF}^\infty_\rho$ formula on a Kripke structure. As before, the variable $X_0$ of the RES, defined initially by the equation $X_0 \stackrel{\nu}{=} \mathsf{AF}_\rho X_0$ is marked in order to indicate that every sequence of dependencies in the boolean graph relating two variables $X_{0j}$ denotes a sequence of transitions matching $\rho$ in the Kripke structure. The symmetric algorithm A4$_{cyc}$ marks all variables $X_{0j}$ and searches for cycles not containing a marked variable: if such a cycle exists, then some sequence satisfying a prefix of $\rho$ in the Kripke structure cannot be extended in a finite number of steps until the end of $\rho$ is reached, and the formula is false. In the example shown on Figure 13, the boolean graph associated to the conjunctive BES contains a cycle (surrounded by the dashed line) that does not contain any marked variable, meaning that the property is false.

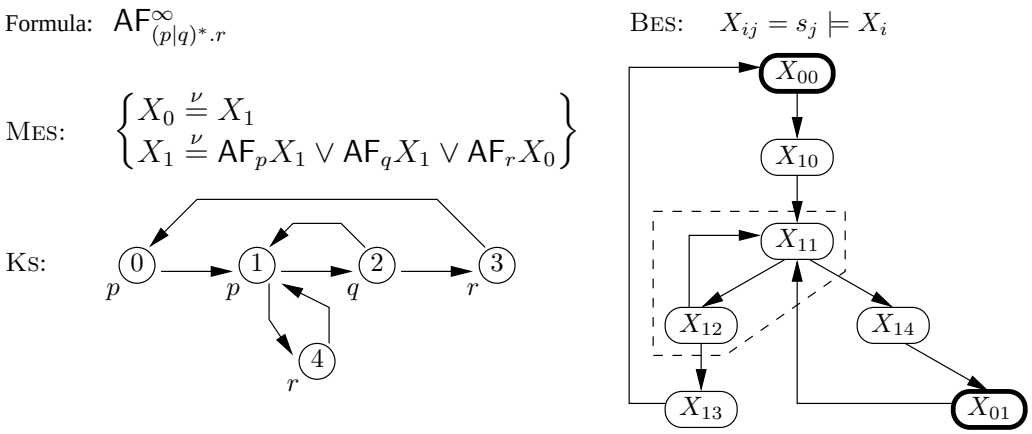


Figure 13: Evaluation of a $\mathsf{AF}^{\infty}$ deterministic formula using the symmetric A4$_{cyc}$ algorithm

The symmetric A4$_{cyc}$ algorithm has a linear time complexity w.r.t. the size of the boolean graph [52], which makes the complexity of evaluating a deterministic $\mathsf{AF}^{\infty}_{\rho}$ formula linear w.r.t. the size of $\rho$ and of the Kripke structure.

**Nondeterministic case.** When $\rho$ is nondeterministic, the BES resulting from the translation above has a general shape, i.e., the right-hand sides of its equations contain both $\vee$ and $\wedge$ connectors. In this case, one can apply the on-the-fly resolution algorithms dedicated to BESs of alternation depth 2 [61]. These algorithms have a worst-case comlexity quadratic in the size of the BES, which yields a quadratic complexity w.r.t. the size of the MES produced by translating $\mathsf{AF}_{\rho}X$ and the size of the Kripke structure.

### 4.3 Complexity

The complexity of the model checking procedure presented in Sections 3, 4.1, and 4.2 is summarized in Table 2. The $\mathsf{EF}_{\rho}$ and $\mathsf{EF}^{\infty}_{\rho}$ operators, together with their respective duals $\mathsf{AG}_{\rho}$ 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. 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 is a consequence of using the memory-efficient BES resolution algorithms A4 [50] and A4$_{cyc}$ [52] dedicated to disjunctive and conjunctive BESs. This fragment of CTRL is the state-based counterpart of PDL-$\Delta$ [57], which is more expressive than CTL$^*$ [26]. 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-$\Delta$ is not succinct [62]. The advantage of the linear-time model checking procedure for the $\mathsf{EF}^{\infty}_{\rho}$ potential looping operator (obtained due to the BES resolution algorithm A4$_{cyc}$ [52]) is to allow an efficient detection of complex cycles in the Kripke structure, which describe oscillation properties [17]. The $\mathsf{EF}^{\infty}_{\rho}$ operator is also

useful for characterizing fairness properties in concurrent systems, such as the existence of complex unfair executions in resource locking protocols [5].

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

| Operator | | Complexity | |
|---|---|---|---|
| | | $\rho$ deterministic | $\rho$ nondeterministic |
| $\mathsf{EF}_\rho$ | $\mathsf{AG}_\rho$ | $O(\|\rho\| \cdot (\|S\| + \|T\|))$ | |
| $\mathsf{AF}_\rho$ | $\mathsf{EG}_\rho$ | $O(\|\rho\| \cdot (\|S\| + \|T\|))$ | $O(2^{\|\rho\|} \cdot (\|S\| + \|T\|))$ |
| $\mathsf{EF}^\infty_\rho$ | $\mathsf{AG}^\dashv_\rho$ | $O(\|\rho\| \cdot (\|S\| + \|T\|))$ | |
| $\mathsf{AF}^\infty_\rho$ | $\mathsf{EG}^\dashv_\rho$ | $O(\|\rho\| \cdot (\|S\| + \|T\|))$ | $O(2^{2\|\rho\|} \cdot (\|S\| + \|T\|)^2)$ |

The $\mathsf{AF}_\rho$ operator and its dual $\mathsf{EG}_\rho$ are evaluated in linear-time only when the regular subformula $\rho$ is deterministic (according to the definition given in Section 3.2.2 in terms of the resulting MES). In the general case, these operators are evaluated in exponential-time w.r.t. the size of $\rho$ (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. The usage of the $\mathsf{AF}_\rho$ operator for specifying properties of protocols and distributed systems often involves regular subformulas $\rho$ with a simple structure (without nested $*$ or $|$ operators, corresponding basically to the nested application of the AF inevitability operators of CTL), which lead to small-sized MESs after translation. Finally, the $\mathsf{AF}^\infty_\rho$ operator and its dual $\mathsf{EG}^\dashv_\rho$ are evaluated in linear-time when $\rho$ is deterministic (thanks to the symmetric version of the $A4_{cyc}$ algorithm); in the general case, these operators are evaluated in doubly exponential-time w.r.t. the size of $\rho$ and in quadratic-time w.r.t. the size of the Kripke structure. This complexity seems difficult to lower, since the BESs produced by translating these operators are of alternation depth 2 and have a general shape (arbitrary nesting of disjunctions and conjunctions in the right-hand sides of equations).

# 5 Implementation and Use

We implemented the model checking procedure for CTRL described in Sections 3 and 4 by reusing as much as possible the on-the-fly verification technology available in the CADP toolbox [33]. This section presents the architecture of our CTRL model checker and illustrates its use for analyzing genetic regulatory networks.

## 5.1 An on-the-fly model checker for CTRL

The most direct way of obtaining a model checker for CTRL is to take advantage of existing verification technology. As verification engine, we use CADP[2] (*Construction and Analysis*

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

*of Distributed Processes*) [33], a state-of-the-art verification toolbox for concurrent asynchronous systems. CADP 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 [39] or CHP [49], as well as networks of communicating automata in the EXP language [45].

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 [31]. CADP contains the on-the-fly model checker EVALUATOR [51], which evaluates regular alternation-free $\mu$-calculus ($L\mu_1^{reg}$) formulas on implicit LTSs. The tool works by translating the verification problem in terms of the local resolution of a BES, which is done using the algorithms available in the generic CÆSAR_SOLVE library [50]. 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 BES resolution. The tool generates full diagnostics (examples and counterexamples) illustrating the truth value of the formulas, and is also equipped with macro-definition mechanisms allowing the creation of reusable libraries of derived temporal operators.

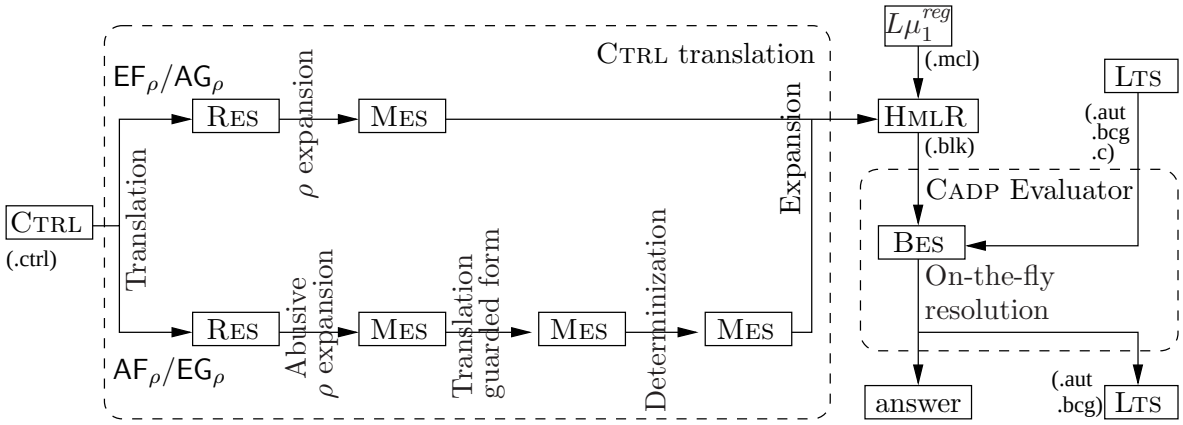


Figure 14: CTRL translator and its connection to the EVALUATOR model checker of CADP

In order to reuse the model checking features of EVALUATOR 3.6, we had the choice of translating CTRL formulas either to $L\mu_1^{reg}$ formulas, or to HMLR specifications. We adopted the second solution because it leads to a more succinct translation and avoids the translation step from $L\mu_1^{reg}$ to HMLR present in EVALUATOR. This technical choice motivated the definition of the translation from CTRL to MESs in the first place. The architecture of the CTRL translator (about 12,000 lines of code) is shown in Figure 14. 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$_\rho$ and AF$_\rho$ 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 shown in Section 4.1. The resulting HMLR specification is directly given as input to EVALUATOR 3.6, together with the LTS corresponding to the Kripke structure.

The translator from CTRL to HMLR has been completely implemented, using the compiler construction technology based upon the SYNTAX[3] system and the LOTOS-NT [34] language proposed in [32], which was successfully used for developing several tools of CADP.

## 5.2 Verification of genetic regulatory networks

CTRL has been used for the analysis of so-called *genetic regulatory networks* (GRNs), which consist of genes, proteins, small molecules and their mutual interactions that together control different functions of the cell. In order to better understand how a specific dynamic behavior emerges from these interactions, and the role of each component in the network, a wide variety of mathematical formalisms are available. The description of the dynamics of GRNs by means of these formalisms results in qualitative or quantitative, discrete or continuous, stochastic or deterministic models [23].

Despite the enormous amount of information accumulated on the components and interactions of GRNs, numerical values for the kinetic parameters and the molecular concentrations are usually absent. As a consequence, the above-mentioned models of GRNs are difficult to apply in practice. This has motivated the use of a special class of *piecewise-linear* (PL) *differential equation* models, originally introduced by [35]. The PL models provide a coarse-grained picture of the dynamics of GRNs. They 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 advantage of using the PL models is that the qualitative dynamics of the high-dimensional systems are relatively straightforward to analyze, using inequality constraints on the parameters rather than exact numerical values [9, 24].

In [9] it is shown how discrete abstractions can be used to convert the continuous dynamics of the PL systems into state transition graphs that are formally equivalent to Kripke structures. The states of the graph correspond to hyperrectangular regions in the concentration space, while the transitions arise from trajectories 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). The generation of the state transition graph from the PL model has been implemented in the computer tool GNA (*Genetic Network Analyzer*) [10]. GNA is able to export the graph to standard model checkers like NUSMV [20] and CADP [33] in order to use formal verification techniques.

We analyse here properties of the carbon starvation response network in *E. coli*, using a PL model proposed in [55] (Figure 15). The dynamics of the system are described by 6 coupled PL differential equations, and 48 inequality constraints on the parameter values. We focus on one particular situation, a nutrient upshift after a period of starvation, leading

[3] SYNTAX is a trademark of INRIA.

to exponential growth of the bacterial population. From a single initial state, we compute the reachable part of the graph consisting of 744 states. The graph contains several cycles, one of which is terminal and corresponds to a (damped) oscillation of some of the protein concentrations and the concentration of stable RNAs.

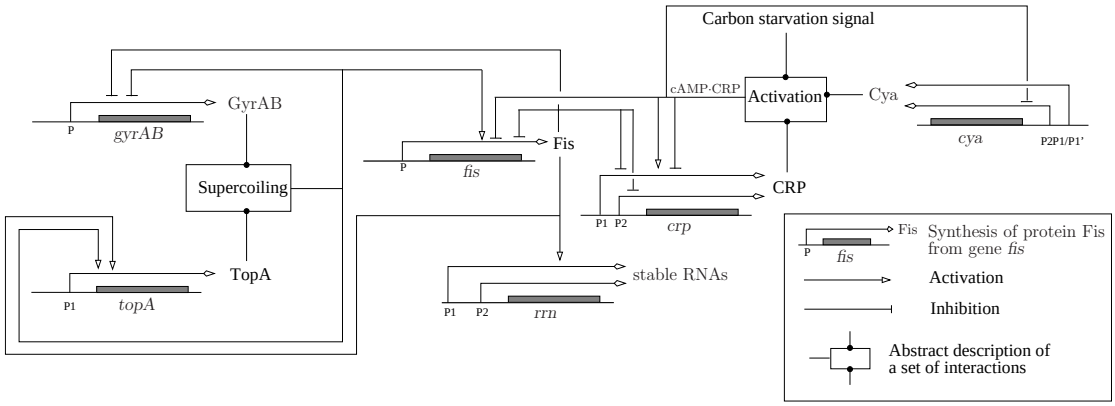


Figure 15: Network of key genes, proteins, and regulatory interactions involved in the GRN controlling the carbon starvation response in *E. coli* [55].

In order to express the latter property in CTRL, we have used a composition of the EF and AF operators. The EF operator does not impose any restrictions on the path, whereas the AF operator stipulates that once the cycle is reached, the concentration of stable RNAs oscillates. This gives rise to the following formula:

$$\mathsf{EF}_{\mathsf{true}^*}\mathsf{AF}_{inTermCycle^+.(inc\_rrn^+.dec\_rrn^+)^+}\mathsf{true} \tag{1}$$

where *inTermCycle* is a predicate denoting that a state is part of the terminal cycle, while *dec_rrn* (*inc_rrn*) represent a decreasing (increasing) concentration of stable RNAs (which are transcribed from the *rrn* operons). The CTRL model checker returns false, indicating that an oscillation of stable RNAs is not inevitable once the system has reached the terminal cycle.

Replacing the AF operator by $\mathsf{EF}^\infty$ results in the following CTRL formula:

$$\mathsf{EF}_{\mathsf{true}^*}\mathsf{EF}^\infty_{inTermCycle^+.(inc\_rrn^+.dec\_rrn^+)^+} \tag{2}$$

The CTRL model checker returns true for this formula. The formula differs from the previous one in two respects. It only specifies that it is possible (instead of necessary) that an oscillation of the concentration of stable RNAs occurs in the terminal cycle. However, it requires the oscillation to continue indefinitely, instead of allowing it to stop after a finite number of periods. We can check a stricter property by asking that all paths in the graph lead to the terminal cycle with an oscillating stable RNA concentration.

$$\mathsf{AG}_{\mathsf{true}^*}\mathsf{EF}^{\infty}_{inTermCycle^+.(inc\_rrn^+.dec\_rrn^+)^+} \tag{3}$$

Formula 3 is false, confirming that some paths do not end up in this cycle. However, we can impose a path restriction on the AG operator, to force the model checker to consider only some of the paths:

$$\mathsf{AG}_{true^*.inc\_Fis^+.dec\_Crp^+.inTermCycle}\mathsf{EF}^{\infty}_{inTermCycle^+.(inc\_rrn^+.dec\_rrn^+)^+} \tag{4}$$

The CTRL model checker returns true for formula 4, proving that all paths satisfying the restriction of an increase of the Fis concentration and a decrease of the Crp concentration, reach the terminal cycle where the concentration of stable RNAs continues to oscillate.

The use of regular expressions in the above CTRL formulas clearly demonstrates the convenience of being able to characterize a sequence of events. Due to the nested iteration operators present in the regular formulas, the CTRL formulas cannot be expressed 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 $\rho$, such as those corresponding to oscillations in the *E. coli* example.

## 6 Conclusions and Future Work

Applications of model checking in system biology have demonstrated its usefulness for understanding the dynamic behaviour of regulatory networks in living cells, but also 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 in order to increase their expressiveness and user-friendliness is a long-standing line of research. One of the first proposals in this direction was ETL [63], 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* [60] and CTL [36]. 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 [4] and Eagle [7] are extensions of Ltl with regular expressions and data handling mechanisms, dedicated respectively to hardware and runtime verification. Rctl [12] is an extension of Ctl with regular expressions, which served subsequently as the basis of the Sugar [11] and Psl [38] specification languages used for hardware verification. RegCtl [14] 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 $\mu$-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 report, 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 properties of the network expressed in Ctrl, 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 [18, 29]. 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 [3, 6, 10, 13, 16, 17, 30].

We plan to continue our work on several directions. First, we will extend the Cæsar_Solve [50] library of Cadp with resolution algorithms handling Bess of alternation depth 2 [61] in order to obtain an on-the-fly evaluation of the $\mathsf{AF}^{\infty}_{\rho}$ operator when the regular formula $\rho$ is nondeterministic. Second, the translation from Ctrl to HmlR can be optimized by adding static analysis features on the Gna atomic propositions in order to reduce the size of the HmlR specifications produced. Third, a distributed version of the Ctrl model checker can be obtained by coupling it with the distributed Bes resolution algorithms proposed in [40, 41]. Fourth, we will develop tools to help non-expert users in applying formal verification to the analysis of biological regulatory networks [53].

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

[2] H. R. Andersen. Model checking and boolean graphs. *Theoretical Computer Science*, 126(1):3–30, April 1994.

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

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

[5] T. Arts, C. Benac Earle, and J. Derrick. Development of a verified Erlang program for resource locking. *Springer International Journal on Software Tools for Technology Transfer (STTT)*, 5(2–3):205–220, March 2004.

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

[7] H. Barringer, A. Goldberg, K. Havelund, and K. Sen. Rule-based runtime verification. In Bernhard Steffen and Giorgio Levi, editors, *Proceedings of the 5th International Conference on Verification, Model Checking, and Abstract Interpretation VMCAI'04 (Venice, Italy)*, volume 2937 of *Lecture Notes in Computer Science*, pages 44–57. Springer Verlag, January 2004.

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

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

[10] G. Batt, D. Ropers, H. de Jong, J. Geiselmann, R. Mateescu, M. Page, and D. Schneider. Validation of qualitative models of genetic regulatory networks by model checking: Analysis of the nutritional stress response in *Escherichia coli*. *Bioinformatics*, 21(Suppl 1):i19–i28, 2005.

[11] I. Beer, S. Ben-David, C. Eisner, D. Fisman, A. Gringauze, and Y. Rodeh. The temporal logic Sugar. In Gérard Berry, Hubert Comon, and Alain Finkel, editors, *Proceedings of the 13th International Conference on Computer Aided Verification CAV'2001 (Paris, France)*, volume 2102 of *Lecture Notes in Computer Science*, pages 363–367. Springer Verlag, July 2001.

[12] I. Beer, S. Ben-David, and A. Landver. On-the-fly model checking of RCTL formulas. In Alan Hu and Moshe Y. Vardi, editors, *Proceedings of the 10th International Conference on Computer Aided Verification CAV'98 (Vancouver, BC, Canada)*, volume 1427 of *Lecture Notes in Computer Science*, pages 184–194. Springer Verlag, June 1998.

[13] G. Bernot, J.-P. Comet, A. Richard, and J. Guespin. Application of formal methods to biological regulatory networks: Extending Thomas' asynchronous logical approach with temporal logic. *Journal of Theoretical Biology*, 229(3):339–348, 2004.

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

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

[16] M. Calder, V. Vyshemirsky, D. Gilbert, and R. Orton. Analysis of signalling pathways using the PRISM model checker. In G. Plotkin, editor, *Computational Methods in Systems Biology, CMSB-05*, pages 79–90, Edinburgh, Scotland, 2005.

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

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

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

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

[21] E. Clarke, O. Grumberg, and D. Peled. *Model Checking*. MIT Press, 2000.

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

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

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

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

[26] E. A. Emerson and J. Y. Halpern. "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)*, pages 127–140, January 1983. Also appeared in Journal of ACM, 33(1):151-178, 1986.

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

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

[29] J. Fisher and T.A. Henzinger. Executable cell biology. *Nature Biotechnology*, 25(11):1239–1250, 2007.

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

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

[32] H. Garavel, F. Lang, and R. Mateescu. Compiler construction using LOTOS NT. In Nigel Horspool, editor, *Proceedings of the 11th International Conference on Compiler Construction CC'2002 (Grenoble, France)*, volume 2304 of *Lecture Notes in Computer Science*, pages 9–13. Springer Verlag, April 2002.

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

[34] H. Garavel and M. Sighireanu. Towards a second generation of formal description techniques – rationale for the design of E-LOTOS. In Jan-Friso Groote, Bas Luttik, and Jos van Wamel, editors, *Proceedings of the 3rd International Workshop on Formal Methods for Industrial Critical Systems FMICS'98 (Amsterdam, The Netherlands)*, pages 187–230, 1998.

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

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

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

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

[39] ISO/IEC. 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, September 1989.

[40] C. Joubert and R. Mateescu. Distributed local resolution of boolean equation systems. In Francisco Tirado and Manuel Prieto, editors, *Proceedings of the 13th Euromicro Conference on Parallel, Distributed and Network-Based Processing PDP'2005 (Lugano, Switzerland)*. IEEE Computer Society, 2005.

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

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

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

[44] D. Kozen. Results on the propositional $\mu$-calculus. *Theoretical Computer Science*, 27:333–354, 1983.

[45] F. Lang. EXP.OPEN 2.0: A flexible tool integrating partial order, compositional, and on-the-fly verification methods. In Jaco van de Pol, Judi Romijn, and Graeme Smith, editors, *Proceedings of the 5th International Conference on Integrated Formal Methods IFM'2005 (Eindhoven, The Netherlands)*, volume 3771 of *Lecture Notes in Computer Science*. Springer Verlag, 2005.

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

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

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

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

[50] R. Mateescu. 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, February 2006. Full version available as INRIA Research Report RR-5948, July 2006.

[51] R. Mateescu and M. Sighireanu. Efficient on-the-fly model-checking for regular alternation-free mu-calculus. *Science of Computer Programming*, 46(3):255–281, March 2003.

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

[53] P.T. Monteiro, D. Ropers, R. Mateescu, A.T. Freitas, and H. de Jong. Temporal logic patterns for querying dynamic models of cellular interaction networks. *Bioinformatics*, 2008. In press.

[54] A. Regev and E. Shapiro. Cells as computation. *Nature*, 419(6905):343, 2002.

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

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

[57] R. Streett. Propositional dynamic logic of looping and converse. *Information and Control*, 1982.

[58] A. Tarski. A lattice-theoretical fixpoint theorem and its applications. *Pacific Journal of Mathematics*, 5, 1955.

[59] R. Thomas, D. Thieffry, and M. Kaufman. Dynamical behaviour of biological regulatory networks: I. Biological role of feedback loops and practical use of the concept of the loop-characteristic state. *Bulletin of Mathematical Biology*, 57(2):247–276, 1995.

[60] W. Thomas. *Computation Tree Logic and regular ω-languages*, volume 354 of *Lecture Notes in Computer Science*, pages 690–713. 1989.

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

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

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

# A Proofs of the translation

## 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 $\delta$ 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 $\delta$.

**Lemma 2** *Let $K$ be a Kripke structure, $B$ an equation block, and $\delta_1, \delta_2$ 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 $\delta_1, \delta_2$ two propositional environments as stated in the hypothesis. The semantics of $B$ in the context of an environment $\delta$ 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 $\Phi_{\delta_1}$ 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 $\varphi$ and any $U_1, ..., U_n \subseteq S$. We proceed by structural induction on $\varphi$.

- $\varphi ::= p$:

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

- $\varphi ::= X$:

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 $X = X_i$.

$$\begin{array}{ll} & [\![X_i]\!]_K((\delta_1 \oslash \delta_2) \oslash [U_1/X_1, ..., U_n/X_n]) \\ = & ((\delta_1 \oslash \delta_2) \oslash [U_1/X_1, ..., U_n/X_n])(X_i) & \text{by def. of } [\![\ ]\!] \\ = & [U_1/X_1, ..., U_n/X_n](X_i) & \text{by def. of } \oslash \\ = & U_i & \text{by def. of } [\ ] \\ = & [\![X_i]\!]_K(\delta_1 \oslash [U_1/X_1, ..., U_n/X_n]) & \text{by def. of } [\![\ ]\!]. \end{array}$$

2. $X \notin \{X_1, ..., X_n\}$, i.e., $X$ is free in $B$. This means $X \in supp(\delta_1)$.

$$\begin{array}{ll} & [\![X]\!]_K((\delta_1 \oslash \delta_2) \oslash [U_1/X_1, ..., U_n/X_n]) \\ = & ((\delta_1 \oslash \delta_2) \oslash [U_1/X_1, ..., U_n/X_n])(X) & \text{by def. of } [\![\ ]\!] \\ = & (\delta_1 \oslash \delta_2)(X) & \text{by def. of } \oslash \\ = & \delta_1(X) & fv(B) \not\subseteq supp(\delta_2) \\ = & [\![X]\!]_K\delta_1 & \text{by def. of } [\![\ ]\!] \\ = & [\![X]\!]_K(\delta_1 \oslash [U_1/X_1, ..., U_n/X_n]) & \text{by def. of } \oslash. \end{array}$$

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

$$\begin{array}{ll} & [\![\varphi_1 \vee \varphi_2]\!]_K((\delta_1 \oslash \delta_2) \oslash [U_1/X_1, ..., U_n/X_n]) \\ = & ([\![\varphi_1]\!]_K((\delta_1 \oslash \delta_2) \oslash [U_1/X_1, ..., U_n/X_n])) \cup \\ & ([\![\varphi_2]\!]_K((\delta_1 \oslash \delta_2) \oslash [U_1/X_1, ..., U_n/X_n])) & \text{by def. of } [\![\ ]\!] \\ = & [\![\varphi_1]\!]_K(\delta_1 \oslash [U_1/X_1, ..., U_n/X_n]) \cup [\![\varphi_2]\!]_K((\delta_1 \oslash [U_1/X_1, ..., U_n/X_n]) & \text{by ind. hyp.} \\ = & [\![\varphi_1 \vee \varphi_2]\!]_K(\delta_1 \oslash [U_1/X_1, ..., U_n/X_n]) & \text{by def. of } [\![\ ]\!]. \end{array}$$

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

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

- $\varphi ::= \mathsf{EF}^\infty_\rho$ (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 $\mathsf{EF}^\infty_\rho$ is closed, so its interpretation is independent of any environment $\delta$.

□

**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 $BL_1$.

- $BL_1 ::= \varepsilon$:

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

- $BL_1 ::= B.BL_1$:

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

□

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

- $\varphi ::= p$:

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

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

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

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

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

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

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

where $\Phi : 2^S \to 2^S$, $\Phi(U) = [\![\mathsf{EF}_\rho X]\!]_K[U/X]$. Note that the $\delta$ environment is omitted in the definition of $\Phi$ 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 [42]:

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

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

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

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

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

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 $\mathsf{EF}^\infty_\rho$, 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 $\Phi$, we have:

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

Based on this, we construct the following path:

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

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 $\delta$. For a given $l \in [1, n]$, the *projection* of $\Phi_\delta$ 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 $\mathsf{EF}_\rho$ and $\mathsf{AG}_\rho$

**Lemma 4** *Let $B = \{X_i \stackrel{\sigma}{=} \varphi_i\}_{1 \leq i \leq n}$ be an equation block, $K$ be a Kripke structure, $\delta$ be an environment, and $\Phi_\delta : (2^S)^n \rightarrow (2^S)^n$ be the functional associated to $B$, $K$, and $\delta$. 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 $\Phi^{l,n}_\delta : (2^S)^{n-l+1} \rightarrow (2^S)^{n-l+1}$ is the projection of $\Phi_\delta$ on the equations $[l, n]$.*

**Proof** Let $B$, $K$, $\delta$, 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 $l \leq j \leq n$:

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

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 [42] 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 $\cup$):

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

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

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

Thus, $(\mu\Phi_\delta)_{[l,n]} = (\bigcup_{k\geq 0} \Phi_\delta^k(\emptyset^n))_{[l,n]} = \bigcup_{k\geq 0}(\Phi_\delta^k(\emptyset^n))_{[l,n]} \sqsubseteq \mu\Phi^{l,n}_{\delta\oslash[(\sigma\Phi_\delta)_1/X_1,\ldots,(\sigma\Phi_\delta)_{l-1}/X_{l-1}]}$, 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 $B = \{X_i \stackrel{\sigma}{=} \varphi_i\}_{1\leq i\leq n}$ be an equation block, and let $\{X_n \stackrel{\sigma}{=} \psi_n, Y_j \stackrel{\sigma}{=} \psi_j\}_{n<j\leq m}$ be another block suitable for the substitution of the equation $X_n \stackrel{\sigma}{=} \varphi_n$ such that $([\![\{X_n \stackrel{\sigma}{=} \varphi_n\}]\!]_K\delta)(X_n) = ([\![\{X_n \stackrel{\sigma}{=} \psi_n, Y_j \stackrel{\sigma}{=} \psi_j\}_{n<j\leq m}]\!]_K\delta)(X_n)$ for any Kripke structure $K$ and environment $\delta$. Then:*

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

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

**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:

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

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 $\Phi_\delta$ associated to $B$ and $\delta$. 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 $\varphi_i$ 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 $[\![\{X_n \overset{\sigma}{=} \psi_n, Y_j \overset{\sigma}{=} \psi_j\}_{n<j\leq m}]\!]_K\delta$, this implies:

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

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

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

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 $\Phi_\delta$.

It remains to show that this is indeed the minimal fixed point of $\Phi_\delta$. 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 [42]:

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

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:

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

For $i = n$, we have:

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

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 $\Phi_\delta$. □

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 $\delta$ 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:

  $$([\![\{X \stackrel{\mu}{=} \mathsf{EF}_{\rho_1.\rho_2}Y\}]\!]_K\delta)(X) = ([\![\{X \stackrel{\mu}{=} \mathsf{EF}_{\rho_1}Z, Z \stackrel{\mu}{=} \mathsf{EF}_{\rho_2}Y\}]\!]_K\delta)(X).$$

  By applying the definition of $[\![\ ]\!]$ and simple properties about substitution of variables in a RES, we obtain:

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

- 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:

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

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

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

- 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:

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

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

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

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

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

  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 [42]:

$$\mu \Phi_\delta = \bigcup_{k \geq 0} \Phi^k(\emptyset), \quad \text{where} \quad \Phi^0(\emptyset) = \emptyset, \quad \Phi^{k+1}(\emptyset) = \delta(Y) \cup \llbracket \mathsf{EF}_{\rho} X \rrbracket_K [\Phi^k(\emptyset)/X].$$

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

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

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.

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

From the above statement, we obtain:

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

□

### A.2.2 Operators $\mathsf{AF}_\rho$ and $\mathsf{EG}_\rho$

**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 $\delta$ 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 [42]:

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

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 $\rho_{ij}$ 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 $\rho_{ij}$ are simply atomic propositions $p_{ij}$.

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

**Proof** Let $K$ be a Kripke structure and $\delta$ 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:

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

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 $\rho$ (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:

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

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 3.2.2. Then:*

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

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

**Proof** Let $K$, $M$, $\delta$, and $P_{\delta,i}$ as stated in the hypothesis. The functional $\Phi_\delta : (2^S)^{2^n-1} \rightarrow (2^S)^{2^n-1}$ 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 [58], 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 $\Phi_\delta$ and the interpretation of $\mathsf{AF}$, 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 3.2.2 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 $\delta$. 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 $\Pi_\delta$ 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 [42]:

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

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.

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

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 $\Pi_\delta$ 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 $\Phi_\delta$, this is equivalent to:

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

Using the definition of $U^k$ and the interpretation of $\mathsf{AF}$, 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 3.2.2 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, $\delta$ 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:

| | |
|---|---|
| $(\llbracket M \rrbracket_K \delta)(X_{\{1\}}) =$ | 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 $\llbracket\ \rrbracket$ |
| $(\llbracket \{X_1 \stackrel{\mu}{=} \mathsf{AF}_\rho Y\} \rrbracket_K \delta)(X_1)$. | |

□

Centre de recherche INRIA Grenoble – Rhône-Alpes
Inovallée, 655, avenue de l'Europe, Montbonnot - 38334 Saint Ismier Cedex (France)

Centre de recherche INRIA Futurs : Parc Club Orsay Université - ZAC des Vignes, 4, rue Jacques Monod - Bât. G - 91893 Orsay Cedex (France)
Centre de recherche INRIA Nancy – Grand Est : 615, rue du Jardin Botanique - 54600 Villers-lès-Nancy (France)
Centre de recherche INRIA Rennes – Bretagne Atlantique : Campus universitaire de Beaulieu - 35042 Rennes Cedex (France)
Centre de recherche INRIA Paris – Rocquencourt : Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex (France)
Centre de recherche INRIA Sophia Antipolis – Méditerranée : 2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis (France)

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-6399