Efficient Diagnostic Generation for Boolean Equation Systems

Radu Mateescu

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Efficient Diagnostic Generation for Boolean Equation Systems

Radu Mateescu

No3861

Janvier 2000

THÈME 1

Rapport de recherche

ISSN 0249-6399 ISRN INRIA/RR--3861--FR+ENG

Efficient Diagnostic Generation for Boolean Equation Systems

Radu Mateescu*

Thème 1 — Réseaux et systèmes
Projet VASY

Rapport de recherche n° 3861 — Janvier 2000 — 23 pages

Abstract: Boolean Equation Systems (Bess) provide a useful framework for the verification of concurrent finite-state systems. In practice, it is desirable that a Bes resolution also yields diagnostic information explaining, preferably in a concise way, the truth value computed for a given variable of the Bes. Using a representation of Bess as extended boolean graphs (Ebgs), we propose a characterization of full diagnostics (i.e., both examples and counterexamples) as a particular class of subgraphs of the Ebg associated to a Bes. We provide algorithms that compute examples and counterexamples in linear time and can be straightforwardly used to extend known (global or local) Bes resolution algorithms with diagnostic generation facilities.

Key-words: boolean equation system, diagnostic, model-checking, mu-calculus, temporal logic, specification, verification

A short version of this report is also available as "Efficient Diagnostic Generation for Boolean Equation Systems," in Susanne Graf, editor, Proceedings of the 3rd International Conference on Tools and Algorithms for the Construction and Analysis of Systems TACAS'2000 (Berlin, Germany), March 2000.

* `Radu.Mateescu@inria.fr`

Génération efficace de diagnostics pour les systèmes d'équations booléennes

Résumé : Les systèmes d'équations booléennes (SEBs) fournissent un cadre utile pour la vérification des systèmes concurrents ayant un nombre fini d'états. En pratique, il est souhaitable que la résolution d'un SEB produise aussi une information de diagnostic qui explique, de préférence de manière concise, la valeur de vérité calculée pour une certaine variable du SEB. Utilisant une représentation des SEBs comme graphes booléens étendus (GBEs), nous proposons une caractérisation des diagnostics complets (c'est-à-dire, exemples et contre-exemples) comme une classe particulière de sous-graphes du GBE associé à un SEB. Nous développons des algorithmes de calcul d'exemples et de contre-exemples ayant une complexité linéaire en taille du GBE (nombre de sommets et d'arcs). Ces algorithmes peuvent être aisément utilisés pour étendre plusieurs algorithmes connus (globaux ou locaux) de résolution d'SEBs avec des facilités de génération de diagnostics.

Mots-clés : diagnostic, logique temporelle, mu-calcul, spécification, système d'équations booléennes, vérification basée sur les modèles

1 Introduction

It is well-known that several equivalence/preorder checking and temporal logic model-checking problems occurring in the verification of concurrent finite-state systems can be reduced to the resolution of Boolean Equation Systems (BESs). Various algorithms have been proposed for solving this problem, either *globally*, i.e., by computing the values of all variables in a BES [3,9,28,1,29,2,20,19], or *locally*, i.e., by computing the value of a single variable [17,1,29,30,20,18,19]. However, practical applications of BES resolution often need more detailed feedback than a simple yes/no answer. For instance, when solving a BES encoding the bisimilarity check between two transition systems, it is desirable to have, in case of a negative result, a *diagnostic* (e.g., a transition sequence) explaining why the two systems are not bisimilar.

In general, both positive diagnostics (examples) and negative diagnostics (counterexamples) are needed in order to be capable of fully explaining the truth value of a boolean variable. This is the case for instance when verifying CTL [5] formulas over a transition system: a positive answer obtained for an $\mathsf{E}\,[\mathsf{T}\ \mathsf{U}\ \varphi]$ formula should be explained by an example (e.g., a transition sequence leading to a φ-state), whereas a negative answer obtained for an $\mathsf{A}\,[\mathsf{T}\ \mathsf{U}\ \varphi]$ formula should be explained by a counterexample (e.g., a transition sequence leading to a deadlock or to a circuit without reaching a φ-state).

The problem of generating diagnostics for finite-state verification has been studied using various approaches. Explicit state enumeration techniques have been applied to compute diagnostics for bisimulation/preorder checking [8,15,13] and CTL model-checking [5,23], in tools like ALDÉBARAN [4] and EMC [5], respectively. Symbolic techniques based on (ordered) binary decision diagrams have been used to generate examples (witnesses) and counterexamples for CTL formulas [6,7], in tools like SMV [21]. Recently, game-based techniques [25] have been applied to verify modal μ-calculus [16] formulas and to interactively generate diagnostics, in tools like the Edinburgh Concurrency Workbench [24].

In this paper we address the problem of characterizing and computing full diagnostics (examples and counterexamples) for BESs. We focus on single fixed point BESs, which allow to encode the alternation-free fragment of the modal μ-calculus [9], and attempt to devise efficient algorithms handling this case. The solutions that we propose can be easily instantiated in order to obtain diagnostic generation facilities for particular verification problems reducible to BES resolution, such as bisimulation/preorder checking and model-checking of branching-time temporal logics like CTL.

We use a representation of BESs as extended boolean graphs (EBGs), which allow to define an appropriate subgraph relation between EBGs. We start by characterizing the solution of a BES by means of two particular temporal logic formulas EX and CX interpreted on the corresponding EBG. This allows, on one hand, to reduce the problem of solving a BES to the problem of verifying these formulas over its EBG and, on the other hand, to characterize minimal diagnostics (w.r.t. the subgraph relation) as particular models of EX or CX. We also propose two efficient (linear-time) algorithms for computing minimal examples and counterexamples and we indicate how they can be used in conjunction with existing (global or local) BES resolution algorithms. Our characterizations of minimal examples and counterexamples turned out to be very similar to the winning strategies for player I and player II of a model-checking game [24]. However, as far as we know, there is no equivalent linear-time complexity result about the game-based algorithms applied to the alternation-free μ-calculus.

The paper is organized as follows. Section 2 defines BESs and their associated EBGs, and gives a characterization of the BES solution using temporal formulas. Section 3 defines diagnostics in terms of subgraphs of an EBG and provides a characterization of minimal diagnostics. Section 4 presents algorithms for computing minimal examples and counterexamples. Finally, Section 5 shows some practical applications of these results and indicates directions for future work.

2 BESs and Extended Boolean Graphs

A *boolean equation system* (BES) M is a set of fixed point equations whose left-hand-sides are boolean variables and whose right-hand-sides are pure disjunctive or conjunctive formulas (see Figure 1). Empty disjunctions and conjunctions are equivalent to F and T, respectively. Variables $\{x_1, ..., x_n\}$ are *bound* and variables in $(\bigcup_{1\leq i\leq n} X_i) \setminus \{x_1, ..., x_n\}$ are *free* in M. A BES is *closed* if it has no free variables. In the sequel, we consider only minimal fixed point BESs ($\sigma = \mu$), the formalization for maximal fixed point BESs being completely dual.

Syntax of Boolean Equation Systems (BESs):

$$M = \{x_i \overset{\sigma}{=} op_i X_i\}_{1\leq i\leq n}$$

where $\sigma \in \{\mu, \nu\}$, $x_i \in \mathcal{X}$, $op_i \in \{\vee, \wedge\}$, $X_i \subseteq \mathcal{X}$ for all $1 \leq i \leq n$

Semantics w.r.t. $\mathbf{Bool} = \{\mathsf{F}, \mathsf{T}\}$ and a context $\delta : \mathcal{X} \to \mathbf{Bool}$:

$$[\![op\{x_1, ..., x_k\}]\!]\,\delta = \delta(x_1)\ op\ ...\ op\ \delta(x_k)$$

$$[\![M]\!]\,\delta = \sigma\Psi_\delta$$

where $\Psi_\delta : \mathbf{Bool}^n \to \mathbf{Bool}^n$, $\Psi_\delta(b_1, ..., b_n) = ([\![op_i X_i]\!]\,\delta[b_1/x_1, ..., b_n/x_n])_{1\leq i\leq n}$

Fig. 1. Syntax and semantics of Boolean Equation Systems

An *extended boolean graph* (EBG) is a tuple $G = (V, E, L, F)$, where: V is the set of vertices; $E \subseteq V \times V$ is the set of edges; $L : V \to \{\vee, \wedge\}$ is the vertex labeling; and $F \subseteq V$ is the *frontier* of G. The notion of frontier will be useful later for defining a suitable subgraph relation between EBGs (see Section 3). The sets of successors and predecessors of a vertex $x \in V$ are noted $E(x)$ and $E^{-1}(x)$, respectively. The set of vertices reachable from x via E is noted $E^*(x)$. The restriction of E to a subset $U \subseteq V$ is defined as $E|_U = \{(x, y) \in E \mid x \in U\}$. Every EBG G induces a Kripke structure $\mathbf{G} = (V, E, L)$. A closed BES can be represented by an EBG, where V denotes the set of boolean variables, E denotes the dependencies between variables, and L labels the vertices as disjunctive or conjunctive according to the operator in the corresponding equation of the Bes.

We can characterize the solution of a closed Bes using temporal logic formulas interpreted over the Kripke structure induced by the corresponding Ebg. The temporal logic we use (see Figure 2) is a simplified variant of the alternation-free modal μ-calculus [10].

Syntax of temporal formulas:

$$\varphi ::= P_\vee \mid P_\wedge \mid \varphi_1 \vee \varphi_2 \mid \varphi_1 \wedge \varphi_2 \mid \langle - \rangle\, \varphi \mid [-]\, \varphi \mid Y \mid \mu Y.\varphi \mid \nu Y.\varphi$$

where $Y \in \mathcal{Y}$

Semantics w.r.t. a Kripke structure $\mathbf{G} = (V, E, L)$ and a context $\rho : \mathcal{Y} \to 2^V$:

$$\begin{aligned}
[\![P_\vee]\!]_{\mathbf{G}}\rho &= \{x \in V \mid L(x) = \vee\} \\
[\![P_\wedge]\!]_{\mathbf{G}}\rho &= \{x \in V \mid L(x) = \wedge\} \\
[\![\varphi_1 \vee \varphi_2]\!]_{\mathbf{G}}\rho &= [\![\varphi_1]\!]_{\mathbf{G}}\rho \cup [\![\varphi_2]\!]_{\mathbf{G}}\rho \\
[\![\varphi_1 \wedge \varphi_2]\!]_{\mathbf{G}}\rho &= [\![\varphi_1]\!]_{\mathbf{G}}\rho \cap [\![\varphi_2]\!]_{\mathbf{G}}\rho \\
[\![\langle - \rangle\, \varphi]\!]_{\mathbf{G}}\rho &= \{x \in V \mid E(x) \cap [\![\varphi]\!]_{\mathbf{G}}\rho \neq \emptyset\} \\
[\![[-]\, \varphi]\!]_{\mathbf{G}}\rho &= \{x \in V \mid E(x) \subseteq [\![\varphi]\!]_{\mathbf{G}}\rho\} \\
[\![Y]\!]_{\mathbf{G}}\rho &= \rho(Y) \\
[\![\mu Y.\varphi]\!]_{\mathbf{G}}\rho &= \bigcap\{U \subseteq V \mid \Phi_{\mathbf{G}\rho}(U) \subseteq U\} \\
[\![\nu Y.\varphi]\!]_{\mathbf{G}}\rho &= \bigcup\{U \subseteq V \mid U \subseteq \Phi_{\mathbf{G}\rho}(U)\}
\end{aligned}$$

where $\Phi_{\mathbf{G}\rho} : 2^V \to 2^V$, $\Phi_{\mathbf{G}\rho}(U) = [\![\varphi]\!]_{\mathbf{G}}\rho[U/Y]$

Fig. 2. Syntax and semantics of the logic for diagnostic characterization

Given a Kripke structure $\mathbf{G} = (V, E, L)$, the two atomic propositions $P_\vee$ and $P_\wedge$ denote the disjunctive and conjunctive vertices of V, respectively. The boolean operators $\vee$ and $\wedge$ have their usual semantics. The possibility and necessity modal formulas $\langle - \rangle\, \varphi$ and $[-]\, \varphi$ denote the vertices for which some (all) successors satisfy φ. The fixed point formulas $\mu Y.\varphi$ and $\nu Y.\varphi$ denote the minimal and maximal solutions (over 2^V) of the equation $Y = \varphi$, respectively. Formulas φ are assumed to be *alternation-free* (without mutual recursion between minimal and maximal fixed points). A vertex $x \in V$ satisfies a formula φ in $\mathbf{G}$, noted $x \models_{\mathbf{G}} \varphi$, iff $x \in [\![\varphi]\!]_{\mathbf{G}}$. $\mathbf{G}$ is a *φ-model* iff $V = [\![\varphi]\!]_{\mathbf{G}}$.

The two particular formulas defined below will be useful in the sequel.

Definition 1 (example and counterexample formulas).
The formulas Ex *and* Cx *defined as follows:*

$$\begin{aligned}
\mathrm{Ex} &= \mu Y.(P_\vee \wedge \langle - \rangle\, Y) \vee (P_\wedge \wedge [-]\, Y) \\
\mathrm{Cx} &= \nu Y.(P_\vee \wedge [-]\, Y) \vee (P_\wedge \wedge \langle - \rangle\, Y)
\end{aligned}$$

are called example formula *and* counterexample formula*, respectively.*

Since Ex and Cx are complementary (Ex $\vee$ Cx $= \mathsf{T}$ and Ex $\wedge$ Cx $= \mathsf{F}$), their interpretations on a Kripke structure $\mathbf{G} = (V, E, L)$ associated to a closed Bes induce a partition of V. The following theorem (proved in Annex A) states that this partition corresponds exactly to the true and false variables in the Bes solution.

Theorem 1 (characterization of BES solution).
Let $M = \{x_i \overset{\mu}{=} op_i X_i\}_{1 \leq i \leq n}$ *be a closed* Bes *and let* $\mathbf{G} = (V, E, L)$ *be its associated Kripke structure. Then:*

$$[\![M]\!]_i = \mathsf{T} \;\Leftrightarrow\; x_i \models_{\mathbf{G}} \text{Ex}$$

for all $1 \leq i \leq n$.

Theorem 1 can be easily extended to alternation-free Bess, whose solution can be characterized using an alternation-free μ-calculus formula containing an Ex-subformula for each single fixed point subsystem[1] of the Bes. The equivalence between alternation-free Bess and alternation-free μ-calculus formulas has been extensively studied in [20]. Together with the classical results of reducing μ-calculus model-checking to Bes resolution [9,1], Theorem 1 provides another proof of this equivalence.

In the following, we will develop the formalization of diagnostics by reasoning exclusively in terms of Ebgs associated to Bess and the interpretations of Ex and Cx formulas on the corresponding Kripke structures.

3 Examples and counterexamples

Consider a Bes M and a boolean variable x that is bound in M. What would be a *diagnostic* for x? From the Bes point of view, a diagnostic for x could be a subsystem M' of M containing x as a bound variable and having the property that by solving M' one obtains for x the same truth value as by solving M. In other words, the value computed for x in M' should not depend upon the context of M' imposed by M (i.e., upon the values of variables that are free in M' and bound in M); that is, it should not depend upon *any* context of M'.

Figure 3 shows a Bes and its associated Ebg, where black vertices denote variables that are T and white vertices denote variables that are F in the Bes solution. According to the informal definition above, a "diagnostic" showing why x_0 is T (an "example" for x_0) would be, for instance, the subsystem defining the variables $\{x_0, x_1, x_2, x_3, x_4\}$, whose vertices are surrounded by a dotted box in the Ebg. Similarly, a "diagnostic" showing why x_5 is F (a "counterexample" for x_5) would be the other subsystem $\{x_5, x_6, x_7, x_8, x_9\}$ outlined in the figure. It is easy to see that these two subsystems can be solved individually and the truth values obtained in this way for x_0 and x_5 are the same as those obtained by solving the whole system.

In general, for a given variable of a Bes there can be several subsystems having the property above (an obvious one being the Bes itself). For instance, the reader may

[1] For ν-subsystems, the formula Ex $= \nu Y.(P_\vee \wedge \langle - \rangle\, Y) \vee (P_\wedge \wedge [-]\, Y)$ must be used.

$$\begin{cases} x_0 \overset{\mu}{=} x_1 \wedge x_4 \\ x_1 \overset{\mu}{=} x_2 \vee x_3 \vee x_5 \\ x_2 \overset{\mu}{=} x_0 \wedge x_1 \\ x_3 \overset{\mu}{=} \mathsf{T} \\ x_4 \overset{\mu}{=} x_1 \vee x_3 \vee x_7 \\ x_5 \overset{\mu}{=} x_6 \wedge x_9 \\ x_6 \overset{\mu}{=} x_3 \wedge x_7 \\ x_7 \overset{\mu}{=} x_3 \wedge x_8 \\ x_8 \overset{\mu}{=} x_4 \wedge x_6 \wedge x_9 \\ x_9 \overset{\mu}{=} \mathsf{F} \end{cases}$$

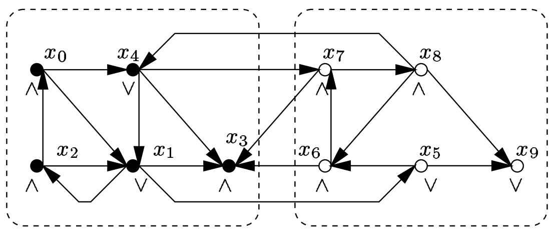

Fig. 3. A closed Bes and its associated Ebg

check that for the Bes on Figure 3, the subsystems $\{x_0, x_1, x_2, x_3, x_4, x_6, x_7, x_8\}$ and $\{x_3, x_4, x_5, x_6, x_7, x_8, x_9\}$ can also be considered as "diagnostics" for the variables x_0 and x_5, respectively.

From the Ebg point of view (and using Theorem 1), a diagnostic for a vertex x of an Ebg G_2 would be a subgraph G_1 of G_2 containing x and having the property that $x \models_{\mathbf{G_1}}$ Ex iff $x \models_{\mathbf{G_2}}$ Ex. A suitable subgraph relation between Ebgs can be defined using the notion of frontier. Intuitively, the frontier of a subgraph G_1 contains all vertices starting at which new edges can be added when G_1 is embedded in another graph G_2 (note that G_2 may have the same vertices as G_1, but more edges). To obtain a correct subgraph relation, the notion of frontier must be *intrinsic* to an Ebg: therefore, when embedding G_1 in G_2, the frontier of G_2 must not contain vertices of G_1 which are not already in the frontier of G_1. The frontier of an Ebg that is not meant to be embedded in another one (e.g., an Ebg associated to a closed Bes) is empty.

Definition 2 (subgraph of an EBG).
Let $G_1 = (V_1, E_1, L_1, F_1)$ and $G_2 = (V_2, E_2, L_2, F_2)$ be two Ebg*s. G_1 is a* subgraph *of G_2, written $G_1 \preceq G_2$, iff the following conditions hold:*

- $V_1 \subseteq V_2$ *and* $F_2 \cap V_1 \subseteq F_1$*;*
- $E_1 \subseteq E_2$ *and* $(E_2 \setminus E_1)|_{V_1} = (E_2 \setminus E_1)|_{F_1}$*;*
- $L_1 = L_2|_{V_1}$.

It is easy to check that $\preceq$ is a partial order relation on Ebgs. For the Ebg on Figure 3, the subgraphs enclosed in the left and right dotted boxes have the frontiers $\{x_1, x_4\}$ and $\{x_6, x_7, x_8\}$, respectively.

The two definitions below precise the notion of diagnostics in terms of EBGs.

Definition 3 (solution-closed EBG).
An EBG $G_1 = (V_1, E_1, L_1, F_1)$ *is* solution-closed *iff, for any* EBG $G_2 = (V_2, E_2, L_2, F_2)$ *such that* $G_1 \preceq G_2$*:*

$$[\![\mathrm{Ex}]\!]_{\mathbf{G_1}} = [\![\mathrm{Ex}]\!]_{\mathbf{G_2}} \cap V_1$$

or, equivalently:

$$[\![\mathrm{Cx}]\!]_{\mathbf{G_1}} = [\![\mathrm{Cx}]\!]_{\mathbf{G_2}} \cap V_1$$

where $\mathbf{G_1}$ *and* $\mathbf{G_2}$ *are the Kripke structures associated to* G_1 *and* G_2*.*

Definition 4 (examples and counterexamples).
Let $G = (V, E, L, F)$ *be an* EBG*,* $\mathbf{G}$ *its associated Kripke structure, and* $x \in V$*. A* diagnostic *for* x *is a solution-closed subgraph of* G *containing* x*. A diagnostic for* x *is called* example *if* $x \models_{\mathbf{G}}$ Ex *and* counterexample *if* $x \models_{\mathbf{G}}$ Cx*.*

The following theorem (proved in Annex A) provides a characterization of solution-closed EBGs that will be useful in the sequel. Intuitively, an EBG G is solution-closed if the satisfaction of Ex (or Cx) on its frontier (which contains the only vertices of G that may directly depend on some external context when G is embedded in another EBG) can be completely decided using only the information in G.

Theorem 2 (characterization of solution-closed EBGs).
Let $G = (V, E, L, F)$ *be an* EBG*.* G *is solution-closed iff:*

$$F \subseteq [\![(P_\vee \wedge \mathrm{Ex}) \vee (P_\wedge \wedge \mathrm{Cx})]\!]_{\mathbf{G}}$$

where $\mathbf{G}$ *is the Kripke structure associated to* G*.*

Using Theorem 2, we can easily see that the left and right subgraphs of the EBG outlined on Figure 3 are solution-closed (i.e., they are diagnostics for x_0 and x_5). The same holds for the subgraphs corresponding to the other two subsystems $\{x_0, x_1, x_2, x_3, x_4, x_6, x_7, x_8\}$ and $\{x_3, x_4, x_5, x_6, x_7, x_8, x_9\}$ having the frontiers $\{x_1, x_8\}$ and $\{x_4\}$. However, in practice it is desirable to explain the value of a variable in a concise manner, and therefore diagnostics should be as small as possible. The following theorem (proved in Annex A) states that *minimal* diagnostics (w.r.t. $\preceq$) can be obtained as particular Ex-models or Cx-models.

Theorem 3 (characterization of minimal diagnostics).
Let $G = (V, E, L, F)$ *be an example for* $x \in V$ *and* $\mathbf{G}$ *its associated Kripke structure.* G *is minimal (w.r.t.* $\preceq$*) iff the following conditions hold:*

a) $\mathbf{G}$ *is an* Ex*-model;*
b) $\forall y \in V.L(y) = \vee \Rightarrow |E(y)| = 1$*;*
c) $V = E^*(x)$*;*
d) $F = \{y \in V \mid L(y) = \vee\}$*.*

The same holds for minimal counterexamples (replacing Ex *by* Cx *and* $\vee$ *by* $\wedge$*).*

The characterization provided by Theorem 3 is sufficiently concrete to allow the design of efficient algorithms for generating minimal diagnostics.

4 Diagnostic generation algorithms

We give in this section algorithms for efficiently computing minimal examples and counterexamples for a given variable of an EBG G by exploring the Kripke structure $\mathbf{G}$ induced by G. These algorithms exploit the information in $[\![\text{Ex}]\!]_{\mathbf{G}}$ and $[\![\text{Cx}]\!]_{\mathbf{G}}$ and therefore they must rely upon a resolution algorithm that first computes the semantics of Ex (or Cx) on $\mathbf{G}$. We start by giving a global resolution algorithm and then we present our diagnostic generation algorithms.

4.1 Global resolution revisited

The global resolution algorithm SOLVE that we consider here (see Figure 4) is a slightly extended version of the global graph-based algorithm given in [1]. The pre- and post-conditions and the invariants of the while-loop are enclosed in rectangular boxes on Figure 4. The SOLVE procedure takes as input a Kripke structure $\mathbf{G} = (V, E, L)$ induced by an EBG G and computes two informations for the vertices $x \in V$: a natural value $c(x)$ such that $c(x) = 0$ iff $x \in [\![\text{Ex}]\!]_{\mathbf{G}}$; and (only for $\vee$-vertices $x \in [\![\text{Ex}]\!]_{\mathbf{G}}$) a successor $s(x) \in E(x)$ such that there is no path from $s(x)$ to x passing only through vertices in $[\![\text{Ex}]\!]_{\mathbf{G}}$.

It is a straightforward exercise to check the validity of the $\mathbf{I_1}$ and $\mathbf{I_2}$ invariants ($\Phi_{\mathbf{G}}^{\text{Ex}}$ is the functional associated to Ex), which ensure that after termination of SOLVE the vertices in $[\![\text{Ex}]\!]_{\mathbf{G}}$ will have $c(x) = 0$. Here we expressed $\mathbf{I_1}$ and $\mathbf{I_2}$ in terms of Ex (we could have done this equivalently in terms of Cx). In the light of Theorem 1, we see that SOLVE is in fact a model-checking algorithm for Ex. This holds also for other global BES resolution algorithms [3,9,28,30].

Invariant $\mathbf{I_3}$ ensures that after termination of SOLVE, all the $\vee$-vertices $x \in [\![\text{Ex}]\!]_{\mathbf{G}}$ will have a successor $s(x) \in [\![\text{Ex}]\!]_{\mathbf{G}}$ such that the satisfaction of Ex by $s(x)$ does not depend upon x. As we will see in the next section, the computation of s is necessary to obtain an efficient algorithm for generating minimal examples.

Figure 5 shows the result of executing SOLVE on the EBG previously considered on Figure 3. Vertices x for which $c(x) = 0$ are black and the others are white. Edges $(x, s(x))$ are drawn as thick arrows.

One can easily adapt other global BES resolution algorithms like those in [3,9,28,30] in order to perform the computation of s. Moreover, we claim that local algorithms like those in [1,29,19] can be adapted as well, since they function by exploring forwards the boolean graph and by propagating backwards the vertices found to be true (which is done in a way similar to the SOLVE algorithm above). In fact, it can be shown that these local algorithms actually compute solution-closed subgraphs containing the boolean variable of interest.

```
$\boxed{\mathbf{G} = (V,\, E,\, L)}$
procedure SOLVE $(V,\, E,\, L)$ is
    forall $x \in V$ do
        $c(x) :=$ if $L(x) = \wedge$ then $|E(x)|$ else $1$ endif
    end;
    $A := \{x \in V \mid c(x) = 0\}$;
    while $A \neq \emptyset$ do                                  $\boxed{\mathbf{I_1} \wedge \mathbf{I_2} \wedge \mathbf{I_3}}$
        let $y \in A$;
        $A := A \setminus \{y\}$;
        forall $z \in E^{-1}(y)$ do
            if $c(z) > 0$ then
                $c(z) := c(z) - 1$;
                if $c(z) = 0$ then
                    $A := A \cup \{z\}$;
                    if $L(z) = \vee$ then
                        $s(z) := y$
                    endif
                endif
            endif
        end
    end
end
$\boxed{\begin{array}{l} \{x \in V \mid c(x) = 0\} = [\![\mathrm{Ex}]\!]_{\mathbf{G}} \wedge \\ \{(x,y) \in E \mid x, y \in [\![\mathrm{Ex}]\!]_{\mathbf{G}} \wedge (L(x) = \vee \Rightarrow y = s(x))\} \text{ is acyclic} \end{array}}$

$\mathbf{I_1} : \Phi_{\mathbf{G}}^{\mathrm{Ex}}(\{x \in V \mid c(x) = 0\} \setminus A) = \{x \in V \mid c(x) = 0\}$
$\mathbf{I_2} : \{x \in V \mid c(x) = 0\} \subseteq \mu\Phi_{\mathbf{G}}^{\mathrm{Ex}} = [\![\mathrm{Ex}]\!]_{\mathbf{G}}$
$\mathbf{I_3} : \{(x,y) \in E \mid c(x) = c(y) = 0 \wedge (L(x) = \vee \Rightarrow y = s(x))\}$ is acyclic
```

Fig. 4. Extended global resolution algorithm

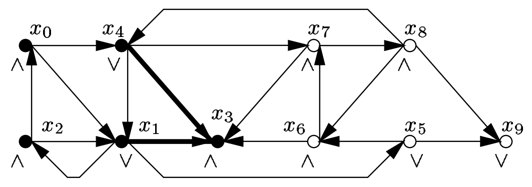

Fig. 5. Computation of c and s by SOLVE

4.2 Generation of minimal examples

The algorithm ExSearch that we propose for computing minimal examples (see Figure 6) takes as input a Kripke structure $\mathbf{G} = (V, E, L)$ induced by an Ebg G, a vertex $x \in [\![\mathrm{Ex}]\!]_{\mathbf{G}}$, and for every $\vee$-vertex $y \in [\![\mathrm{Ex}]\!]_{\mathbf{G}}$ a successor $s(y)$ as computed by the Solve algorithm given in Section 4.1. Note that the algorithm does not explicitly need *all* the vertices in $[\![\mathrm{Ex}]\!]_{\mathbf{G}}$ but only the fact that $x \in [\![\mathrm{Ex}]\!]_{\mathbf{G}}$ and the information given by s.

ExSearch iteratively accumulates in V_0 all the vertices in $[\![\mathrm{Ex}]\!]_{\mathbf{G}}$ that are reachable from x by traversing only edges $(y, s(y))$ if $L(y) = \vee$ and edges $(y, z) \in E$ if $L(y) = \wedge$. All traversed edges are accumulated in E_0.

Invariant $\mathbf{J_1}$ (ensured by the properties of s) implies that after termination of ExSearch, $\mathbf{G_0} = (V_0, E_0, L|_{V_0})$ is an Ex-model. Indeed, at the end of the while-loop $A = \emptyset$ and thus $V_0 \subseteq \bigcup_{i \geq 0} {\Phi^{\mathrm{Ex}}_{\mathbf{G_0}}}^{i}(\emptyset) = \mu \Phi^{\mathrm{Ex}}_{\mathbf{G_0}} = [\![\mathrm{Ex}]\!]_{\mathbf{G_0}} \subseteq V_0$. Invariant $\mathbf{J_2}$ implies that all $\vee$-vertices $y \in V_0$ have only one successor (namely $s(y)$), and invariant $\mathbf{J_3}$ implies that all vertices in V_0 are reachable from x via E_0. $\mathbf{G_0}$ being an Ex-model, Theorem 2 ensures that G_0 is solution-closed, i.e., it is an example for x. Moreover, G_0 meets the conditions of Theorem 3 and thus it is minimal.

Figure 7 shows a minimal example G_0 computed by ExSearch for the variable x_0 in the Ebg considered earlier on Figure 5. The edges in E_0 are drawn as thick arrows and the vertices on the frontier of G_0 are surrounded by dashed circles. The $\vee$-vertices x_1 and x_4 have in E_0 a unique successor $s(x_1) = s(x_4) = x_3$ that was previously computed by Solve.

Note that the use of the information in s is crucial for ensuring the correctness of ExSearch: if we chose for x_1 the successor x_2 instead of x_3, the algorithm would compute the subgraph G_0 outlined on Figure 8, which is *not* an example for x_0 because $x_0 \models_{\mathbf{G}_0} \mathrm{Cx}$. A correct version of ExSearch that does not use s would require a backtracking graph search algorithm in order to determine the "good" successor for each $\vee$-vertex of the example. It is not obvious how to obtain a linear-time algorithm for computing minimal examples in this way.

ExSearch has a complexity $O(|V_0| + |E_0|)$, since all vertices (edges) in the constructed example G_0 are visited (traversed) only once. Since this is the lowest possible complexity for an algorithm that must entirely explore G_0, it appears that (modulo the linear-time precomputation of s) ExSearch is an optimal algorithm for finding minimal examples. In practice, ExSearch runs very quickly when computing examples whose sizes are significantly smaller than $[\![\mathrm{Ex}]\!]_{\mathbf{G}}$ (this happens for Ctl formulas like $\mathsf{E}\,[\mathsf{T}\ \mathsf{U}\ \varphi]$).

$\boxed{\begin{array}{l}\mathbf{G} = (V, E, L) \wedge x \in [\![\mathrm{Ex}]\!]_{\mathbf{G}} \wedge \\ R = \{(y, z) \in E \mid y, z \in [\![\mathrm{Ex}]\!]_{\mathbf{G}} \wedge (L(y) = \vee \Rightarrow z = s(y))\} \text{ is acyclic}\end{array}}$

```
procedure ExSearch ($x$, ($V$, $E$, $L$), $s$) is
    $V_0 := \{x\}$; $E_0 := \emptyset$; $A := \{x\}$;
    while $A \neq \emptyset$ do                                   $\boxed{\mathbf{J_1} \wedge \mathbf{J_2} \wedge \mathbf{J_3}}$
        let $y \in A$;
        $A := A \setminus \{y\}$;
        if $L(y) = \vee$ then
            $E_0 := E_0 \cup \{(y, s(y))\}$;
            if $s(y) \notin V_0$ then
                $V_0 := V_0 \cup \{s(y)\}$; $A := A \cup \{s(y)\}$
            endif
        else
            forall $z \in E(y)$ do
                $E_0 := E_0 \cup \{(y, z)\}$;
                if $z \notin V_0$ then
                    $V_0 := V_0 \cup \{z\}$; $A := A \cup \{z\}$
                endif
            end
        endif
    end
end
```

$\boxed{G_0 = (V_0, E_0, L|_{V_0}, \{y \in V_0 \mid L(y) = \vee\}) \text{ is a minimal example for } x}$

$\mathbf{J_1} : \exists k \geq 0.(V_0 \subseteq \bigcup_{i=0}^{k} {\Phi^{\mathrm{Ex}}_{(V_0, E_0, L|_{V_0})}}^{i}(A))$
$\mathbf{J_2} : E_0 = R|_{V_0}$
$\mathbf{J_3} : V_0 = E_0^*(x)$

Fig. 6. Minimal example generation algorithm

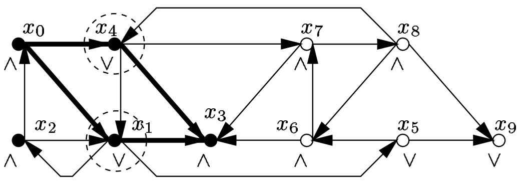

Fig. 7. A minimal example for x_0 computed by ExSearch

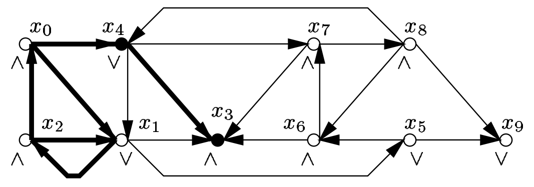

Fig. 8. An erroneous example for x_0 computed in absence of s

4.3 Generation of minimal counterexamples

The algorithm CxSearch that we propose for computing minimal counterexamples (see Figure 9) takes as input a Kripke structure $\mathbf{G} = (V, E, L)$ induced by an Ebg G, a vertex $x \in [\![\mathrm{Cx}]\!]_{\mathbf{G}}$, and for every vertex $y \in V$ a counter $c(y)$ as computed by the Solve algorithm given in Section 4.1.

CxSearch iteratively accumulates in V_0 all the vertices in $[\![\mathrm{Cx}]\!]_{\mathbf{G}}$ that are reachable from x by traversing either a single edge $(y, z) \in E$ if $L(y) = \wedge$, or all edges $(y, z) \in E$ if $L(y) = \vee$. All traversed edges are accumulated in E_0.

Invariant $\mathbf{K_1}$ ($\Phi_{\mathbf{G}}^{\mathrm{Cx}}$ is the functional associated to Cx) ensures that after termination of CxSearch, $\mathbf{G_0} = (V_0, E_0, L|_{V_0})$ is a Cx-model. Indeed, at the end of the while-loop $A = \emptyset$ and thus $V_0 \subseteq \Phi_{\mathbf{G_0}}^{\mathrm{Cx}}(V_0)$. By Tarski's theorem [27], this implies $V_0 \subseteq \nu\Phi_{\mathbf{G_0}}^{\mathrm{Cx}} = [\![\mathrm{Cx}]\!]_{\mathbf{G_0}} \subseteq V_0$. Invariant $\mathbf{K_2}$ implies that after the while-loop $\wedge$-vertices of V_0 have only one successor in V_0 and $\vee$-vertices have all their successors in V_0. Invariant $\mathbf{K_3}$ implies that all vertices in V_0 are reachable from x via E_0. Since $\mathbf{G_0}$ is a Cx-model, Theorem 2 ensures that G_0 is solution-closed, i.e., it is a counterexample for x. Moreover, G_0 meets the conditions of Theorem 3 and thus it is minimal.

Figure 10 shows a minimal counterexample G_0 computed by CxSearch for the variable x_5 in the Ebg considered earlier on Figure 5.

CxSearch has a complexity $O(|V_0| + |E_0|)$, since all vertices (edges) in the constructed counterexample G_0 are visited (traversed) only once. Since this is the lowest possible complexity for an algorithm that must entirely explore G_0, CxSearch appears to be an optimal algorithm for finding minimal counterexamples. In practice, CxSearch runs very quickly when computing counterexamples whose sizes are significantly smaller than $[\![\mathrm{Cx}]\!]_{\mathbf{G}}$ (this happens for Ctl formulas like $\mathsf{A}\,[\mathsf{T}\ \mathsf{U}\ \varphi]$).

$\boxed{\mathbf{G} = (V, E, L) \wedge x \in [\![\mathrm{Cx}]\!]_{\mathbf{G}} = \{y \in V \mid c(y) > 0\}}$

```
procedure CxSearch ($x$, ($V$, $E$, $L$), $c$) is
  $V_0 := \{x\}$; $E_0 := \emptyset$; $A := \{x\}$;
  while $A \neq \emptyset$ do                                    $\boxed{\mathbf{K_1} \wedge \mathbf{K_2} \wedge \mathbf{K_3}}$
    let $y \in A$;
    $A := A \setminus \{y\}$;
    if $L(y) = \wedge$ then
      let $z \in E(y)$ such that $c(z) > 0$;
      $E_0 := E_0 \cup \{(y, z)\}$;
      if $z \notin V_0$ then
        $V_0 := V_0 \cup \{z\}$; $A := A \cup \{z\}$
      endif
    else
      forall $z \in E(y)$ do
        $E_0 := E_0 \cup \{(y, z)\}$;
        if $z \notin V_0$ then
          $V_0 := V_0 \cup \{z\}$; $A := A \cup \{z\}$
        endif
      end
    endif
  end
end
```

$\boxed{G_0 = (V_0, E_0, L|_{V_0}, \{y \in V_0 \mid L(y) = \wedge\}) \text{ is a minimal counterexample for } x}$

$\mathbf{K_1} : V_0 \setminus A \subseteq \Phi^{\mathrm{Cx}}_{(V_0, E_0, L|_{V_0})}(V_0)$

$\mathbf{K_2} : \forall y \in V_0 \setminus A.(L(y) = \wedge \Rightarrow |E_0(y)| = 1) \wedge (L(y) = \vee \Rightarrow |E_0(y)| = |E(y)|)$

$\mathbf{K_3} : V_0 = E_0{}^*(x)$

Fig. 9. Minimal counterexample generation algorithm

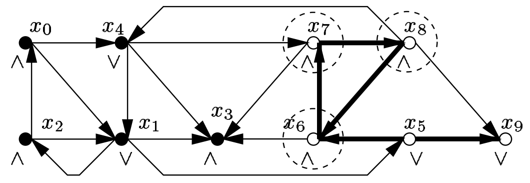

Fig. 10. A minimal counterexample for x_5 computed by CxSearch

5 Conclusion and future work

By representing a boolean equation system M as an extended boolean graph G, we characterized the solution of M by means of two particular alternation-free μ-calculus formulas Ex and Cx interpreted on the Kripke structure $\mathbf{G}$ induced by G. This allowed to identify full diagnostics (examples and counterexamples) explaining the truth value of a boolean variable x of M as being particular subgraphs of G containing x. Moreover, minimal examples and counterexamples (w.r.t. a subgraph relation that we defined) are obtained as particular models of Ex and Cx, respectively.

The temporal logic-based formalization that we proposed provides a uniform framework for analyzing graph-based Bes resolution algorithms such as those in [3,9,28,1,19]. For instance, in Section 4.1 we used our formalization to prove the correctness of a global resolution algorithm from [1], which can be seen in fact as an algorithm for checking the Ex formula on a boolean graph.

We presented two linear-time algorithms ExSearch and CxSearch that compute minimal examples and counterexamples for a given variable of a Bes. We also indicated how these algorithms can be used to extend existing (global or local) Bes resolution algorithms with diagnostic generation facilities.

These two algorithms have been included in the model-checker Evaluator version 3.0 that we developed as part of the Cadp (Cæsar/Aldébaran) protocol engineering toolset [11] using the generic Open/Cæsar environment for on-the-fly verification [14]. Evaluator 3.0 performs on-the-fly model-checking of alternation-free μ-calculus formulas extended with regular expressions as in Pdl-Δ [26]. The diagnostic generation facilities proved to be extremely useful in practice, as illustrated by the use of the model-checker by non-expert users and also for teaching purposes. Besides giving diagnostics for plain alternation-free μ-calculus formulas, Evaluator 3.0 can be used to find regular execution sequences in labeled transition systems (as diagnostics for Pdl-Δ formulas) and to produce full diagnostics for Ctl [5] and Actl [22] formulas (by encoding the operators of these logics as macro-definitions in the input language of the tool).

The ExSearch and CxSearch algorithms compute diagnostics that are minimal w.r.t. the Ebg subgraph relation that we proposed. The diagnostics obtained contain no redundant information, since every $\vee$-vertex in a minimal example and every $\wedge$-vertex in a minimal counterexample has only one successor. This is reasonably good in practice, as confirmed by the experiments performed using Evaluator 3.0. However, there are other additional criteria that may be considered for further reducing the diagnostic size (e.g., minimizing the number of vertices, number of edges, depth, diameter, etc.). Some of these optimizations can be done efficiently in particular cases, e.g., generating minimal length transition sequences as diagnostics for Pdl-Δ diamond modalities or Ctl formulas $\mathsf{E}\,[\mathsf{T}\ \mathsf{U}\ \varphi]$ (which both translate into Bess containing only $\vee$ operators in the nontrivial right-hand sides). An interesting issue would be to investigate the general extension of ExSearch and CxSearch with such optimization features.

We also plan to apply our diagnostic generation techniques in the context of bisimulation checking [9,2] and of test generation [12]. Another potentially fruitful direction of research is to extend our formalization to BESs of higher alternation depth [29,2,20,18]. The characterizations of the solution and diagnostics for these BESs would certainly require formulas of the full modal μ-calculus.

Acknowledgements

We are grateful to the anonymous referees for their valuable comments and suggestions. We also thank Mihaela Sighireanu for largely contributing to the design and implementation of the EVALUATOR version 3.0 model-checker.

References

1. H. R. Andersen. Model checking and boolean graphs. *Theoretical Computer Science*, 126(1):3–30, April 1994.
2. H. R. Andersen and B. Vergauwen. Efficient Checking of Behavioural Relations and Modal Assertions using Fixed-Point Inversion. In P. Wolper, editor, *Proceedings of the 7th International Conference on Computer Aided Verification CAV '95 (Liege, Belgium)*, volume 939 of *Lecture Notes in Computer Science*, pages 142–154. Springer Verlag, July 1995.
3. A. Arnold and P. Crubillé. A linear algorithm to solve fixed-point equations on transition systems. *Information Processing Letters*, 29:57–66, 1988.
4. M. Bozga, J-C. Fernandez, A. Kerbrat, and L. Mounier. Protocol Verification with the ALDEBARAN toolset. *Springer International Journal on Software Tools for Technology Transfer (STTT)*, 1(1-2):166–183, 1997.
5. E. M. Clarke, E. A. Emerson, and A. P. Sistla. Automatic Verification of Finite-State Concurrent Systems using Temporal Logic Specifications. *ACM Transactions on Programming Languages and Systems*, 8(2):244–263, April 1986.
6. E. M. Clarke, O. Grumberg, and D. Long. Verification Tools for Finite-State Concurrent Systems. In J. W. de Bakker, W-P. de Roever, and G. Rozenberg, editors, *Proceedings of the REX School/Symposium "A Decade of Concurrency - Reflections and Perspectives" (Noordwijkerhout, The Netherlands)*, volume 803 of *Lecture Notes in Computer Science*, pages 124–175, Berlin, June 1993. Springer Verlag.
7. E. M. Clarke, O. Grumberg, K. L. McMillan, and X. Zhao. Efficient Generation of Counterexamples and Witnesses in Symbolic Model Checking. In *Proceedings of the 32nd Design Automation Conference DAC'95 (San Francisco, CA, USA)*, pages 427–432. ACM, June 1995.
8. R. Cleaveland. On Automatically Explaining Bisimulation Inequivalence. 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 364–372, Berlin, June 1990. Springer Verlag.
9. R. Cleaveland and B. Steffen. A Linear-Time Model-Checking Algorithm for the Alternation-Free Modal Mu-Calculus. In K. G. Larsen and A. Skou, editors, *Proceedings of 3rd Workshop on Computer Aided Verification CAV '91 (Aalborg, Denmark)*, volume 575 of *Lecture Notes in Computer Science*, pages 48–58, Berlin, July 1991. Springer Verlag.

10. E. A. Emerson and C-L. Lei. Efficient Model Checking in Fragments of the Propositional Mu-Calculus. In *Proceedings of the 1st LICS*, pages 267–278, 1986.
11. Jean-Claude Fernandez, Hubert Garavel, Alain Kerbrat, Radu Mateescu, Laurent Mounier, and Mihaela Sighireanu. CADP (CÆSAR/ALDEBARAN Development Package): A Protocol Validation and Verification Toolbox. In Rajeev Alur and Thomas A. Henzinger, editors, *Proceedings of the 8th Conference on Computer-Aided Verification (New Brunswick, New Jersey, USA)*, volume 1102 of *Lecture Notes in Computer Science*, pages 437–440. Springer Verlag, August 1996.
12. Jean-Claude Fernandez, Claude Jard, Thierry Jéron, Laurence Nedelka, and César Viho. Using On-the-Fly Verification Techniques for the Generation of Test Suites. In R. Alur and T. A. Henzinger, editors, *Proceedings of the 8th International Conference on Computer-Aided Verification (Rutgers University, New Brunswick, NJ, USA)*, volume 1102 of *Lecture Notes in Computer Science*, pages 348–359. Springer Verlag, August 1996. Also available as INRIA Research Report RR-2987.
13. Jean-Claude Fernandez and Laurent Mounier. "On the Fly" Verification of Behavioural Equivalences and Preorders. In K. G. Larsen and A. Skou, editors, *Proceedings of the 3rd Workshop on Computer-Aided Verification (Aalborg, Denmark)*, volume 575 of *Lecture Notes in Computer Science*, Berlin, July 1991. Springer Verlag.
14. Hubert 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.
15. H. Korver. Computing Distinguishing Formulas for Branching Bisimulation. In K. G. Larsen and A. Skou, editors, *Proceedings of the 3rd International Workshop CAV '91 (Aalborg, Denmark)*, volume 575 of *Lecture Notes in Computer Science*, pages 13–23, Berlin, July 1991. Springer Verlag.
16. D. Kozen. Results on the Propositional μ-calculus. *Theoretical Computer Science*, 27:333–354, 1983.
17. K. G. Larsen. Efficient Local Correctness Checking. In G. v. Bochmann and D. K. Probst, editors, *Proceedings of 4th International Workshop in Computer Aided Verification CAV '92 (Montréal, Canada)*, volume 663 of *Lecture Notes in Computer Science*, pages 30–43, Berlin, June-July 1992. Springer Verlag.
18. X. Liu, C. R. Ramakrishnan, and S. A. Smolka. Fully Local and Efficient Evaluation of Alternating Fixed Points. In Bernhard Steffen, editor, *Proceedings of 1st 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 5–19, Berlin, March 1998. Springer Verlag.
19. X. Liu and S. A. Smolka. Simple Linear-Time Algorithms for Minimal Fixed Points. In Kim G. Larsen, Sven Skyum, and Glynn Winskel, editors, *Proceedings of the 25th International Colloquium on Automata, Languages, and Programming ICALP'98 (Aalborg, Denmark)*, volume 1443 of *Lecture Notes in Computer Science*, pages 53–66. Springer Verlag, July 1998.
20. Angelika Mader. *Verification of Modal Properties Using Boolean Equation Systems.* VERSAL 8, Bertz Verlag, Berlin, 1997.
21. K. L. McMillan. *Symbolic Model Checking.* Kluwer Academic Publishers, 1993.

22. R. De Nicola and F. W. Vaandrager. *Action versus State based Logics for Transition Systems.* In *Proceedings Ecole de Printemps on Semantics of Concurrency*, volume 469 of *Lecture Notes in Computer Science*, pages 407–419. Springer Verlag, 1990.
23. A. Rasse. Error diagnosis in finite communicating systems. In K. G. Larsen and A. Skou, editors, *Proceedings of 3rd Workshop on Computer Aided Verification CAV '91 (Aalborg, Denmark)*, volume 575 of *Lecture Notes in Computer Science*, pages 114–124, Berlin, July 1991. Springer Verlag.
24. Perdita Stevens and Colin Stirling. Practical Model-Checking using Games. 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 85–101, Berlin, March 1998. Springer Verlag.
25. C. Stirling. Bisimulation, model checking and other games. In *Notes for Mathfit instructional meeting on games and computation*, Edinburgh, June 1997.
26. R. Streett. Propositional Dynamic Logic of Looping and Converse. *Information and Control*, (54):121–141, 1982.
27. A. Tarski. A Lattice-Theoretical Fixpoint Theorem and its Applications. *Pacific Journal of Mathematics*, (5):285–309, 1955.
28. B. Vergauwen and J. Lewi. A linear algorithm for solving fixed-point equations on transition systems. In *Proceedings of the 17th Colloquium on Trees in Algebra and Programming CAAP '92 (Rennes, France)*, volume 581 of *Lecture Notes in Computer Science*, pages 322–341, Berlin, February 1992. Springer Verlag.
29. 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.
30. B. Vergauwen, J. Wauman, and J. Lewi. Efficient FixPoint Computation. In *Proceedings of the 1st International Static Analysis Symposium SAS '94 (Namur, Belgium)*, volume 864 of *Lecture Notes in Computer Science*, pages 314–328, Berlin, September 1994. Springer Verlag.

A Proofs

We first define some useful shorthand notations. Consider a Kripke structure $\mathbf{G} = (V, E, L)$. According to Definition 1 and to Figure 2, the functionals $\Phi_{\mathbf{G}}^{\mathrm{Ex}}, \Phi_{\mathbf{G}}^{\mathrm{Cx}} : 2^V \to 2^V$ associated to the Ex and Cx formulas are expressed below:

$$\begin{aligned}
\Phi_{\mathbf{G}}^{\mathrm{Ex}}(U) &= [\![(P_\vee \wedge \langle - \rangle Y) \vee (P_\wedge \wedge [-]\, Y)]\!]_{\mathbf{G}}[U/Y] \\
&= \{x \in V \mid ((L(x) = \vee) \wedge E(x) \cap U \neq \emptyset) \vee ((L(x) = \wedge) \wedge E(x) \subseteq U)\} \\
\Phi_{\mathbf{G}}^{\mathrm{Cx}}(U) &= [\![(P_\vee \wedge [-]\, Y) \vee (P_\wedge \wedge \langle - \rangle Y)]\!]_{\mathbf{G}}[U/Y] \\
&= \{x \in V \mid ((L(x) = \vee) \wedge E(x) \subseteq U) \vee ((L(x) = \wedge) \wedge E(x) \cap U \neq \emptyset)\}
\end{aligned}$$

The semantics of Ex and Cx over $\mathbf{G}$ are $[\![\mathrm{Ex}]\!]_{\mathbf{G}} = \mu \Phi_{\mathbf{G}}^{\mathrm{Ex}}$ and $[\![\mathrm{Cx}]\!]_{\mathbf{G}} = \nu \Phi_{\mathbf{G}}^{\mathrm{Cx}}$. Let $\mathrm{Ex}_{\mathbf{G}}^k$ be the increasing chain defined by $\mathrm{Ex}_{\mathbf{G}}^{k+1} = \Phi_{\mathbf{G}}^{\mathrm{Ex}}(\mathrm{Ex}_{\mathbf{G}}^k)$, $\mathrm{Ex}_{\mathbf{G}}^0 = \emptyset$ and $\mathrm{Cx}_{\mathbf{G}}^k$ be the decreasing chain defined by $\mathrm{Cx}_{\mathbf{G}}^{k+1} = \Phi_{\mathbf{G}}^{\mathrm{CX}}(\mathrm{Cx}_{\mathbf{G}}^k)$, $\mathrm{Cx}_{\mathbf{G}}^0 = V$. Using the Knaster-Tarski theorem, the semantics of Ex and Cx over $\mathbf{G}$ can be obtained as $[\![\mathrm{Ex}]\!]_{\mathbf{G}} = \mu \Phi_{\mathbf{G}}^{\mathrm{Ex}} = \bigcup_{k \geq 0} \mathrm{Ex}_{\mathbf{G}}^k$ and $[\![\mathrm{Cx}]\!]_{\mathbf{G}} = \nu \Phi_{\mathbf{G}}^{\mathrm{Cx}} = \bigcap_{k \geq 0} \mathrm{Cx}_{\mathbf{G}}^k$, respectively.

Proof (Theorem 1).
Consider a closed BES $M = \{x_i \overset{\mu}{=} op_i X_i\}_{1 \le i \le n}$ and its associated Kripke structure $\mathbf{G} = (V, E, L)$. According to classical μ-calculus model-checking results [3,9,28,1], we can translate the satisfaction of a single fixed point formula $\sigma Y.\varphi$ by a vertex $x_i \in V$ $(1 \le i \le n)$ into the resolution of a BES N as follows:

$$x_i \models_{\mathbf{G}} \sigma Y.\varphi \;\Leftrightarrow\; [\![N]\!]_i = \mathsf{T}$$

where $N = \{Y_{x_i} \overset{\sigma}{=} (\varphi)_{x_i}\}_{1 \le i \le n}$ and $(\varphi)_x$ is obtained using the rules (1) below.

$$\boxed{\begin{aligned}(P_\vee)_x &= (L(x) = \vee)\\ (P_\wedge)_x &= (L(x) = \wedge)\\ (\varphi_1 \vee \varphi_2)_x &= (\varphi_1)_x \vee (\varphi_2)_x\\ (\varphi_1 \wedge \varphi_2)_x &= (\varphi_1)_x \wedge (\varphi_2)_x\\ (\langle - \rangle\, \varphi)_x &= \textstyle\bigvee_{x' \in E(x)} (\varphi)_{x'}\\ ([-]\, \varphi)_x &= \textstyle\bigwedge_{x' \in E(x)} (\varphi)_{x'}\end{aligned}} \tag{1}$$

By applying this translation to EX interpreted on $\mathbf{G}$, we obtain, for all $1 \le i \le n$:

$$x_i \models_{\mathbf{G}} \mu Y.(P_\vee \wedge \langle - \rangle\, Y) \vee (P_\wedge \wedge [-]\, Y) \;\Leftrightarrow\; [\![N]\!]_i = \mathsf{T}$$

where:

$$\begin{aligned}N &= \{Y_{x_i} \overset{\mu}{=} ((P_\vee \wedge \langle - \rangle\, Y) \vee (P_\wedge \wedge [-]\, Y))_{x_i}\}_{1 \le i \le n}\\ &= \left\{ Y_{x_i} \overset{\mu}{=} \begin{cases} \bigvee_{x_j \in E(x_i)} Y_{x_j} & \text{if } L(x_i) = \vee \\ \bigwedge_{x_j \in E(x_i)} Y_{x_j} & \text{if } L(x_i) = \wedge \end{cases} \right\}_{1 \le i \le n} && \text{by using rules (1)}\\ &= \{Y_{x_i} \overset{\mu}{=} op_i \{Y_{x_j} \mid x_j \in X_i\}\}_{1 \le i \le n} && \text{by definition of } \mathbf{G}\\ &= M && \text{by renaming } Y_{x_i} \text{ into } x_i \ (1 \le i \le n).\end{aligned}$$

Proof (Theorem 2).

If. Let $G = (V, E, L, F)$ such that $F \subseteq [\![(P_\vee \wedge \text{EX}) \vee (P_\wedge \wedge \text{CX})]\!]_{\mathbf{G}}$ and let $G' = (V', E', L', F')$ such that $G \preceq G'$. We must show that $[\![\text{EX}]\!]_{\mathbf{G}} = [\![\text{EX}]\!]_{\mathbf{G}'} \cap V$. Using the complementarity between EX and CX, we can split this equality into $[\![\text{EX}]\!]_{\mathbf{G}'} \cap V \subseteq [\![\text{EX}]\!]_{\mathbf{G}}$ and $[\![\text{CX}]\!]_{\mathbf{G}'} \cap V \subseteq [\![\text{CX}]\!]_{\mathbf{G}}$.

- To prove $[\![\text{EX}]\!]_{\mathbf{G}'} \cap V \subseteq [\![\text{EX}]\!]_{\mathbf{G}}$, we show by induction that $\text{EX}^k_{\mathbf{G}'} \cap V \subseteq [\![\text{EX}]\!]_{\mathbf{G}}$ for all $k \ge 0$.
  Base step. $\text{EX}^0_{\mathbf{G}'} \cap V = \emptyset \subseteq [\![\text{EX}]\!]_{\mathbf{G}}$.
  Inductive step. Let $x \in \text{EX}^{k+1}_{\mathbf{G}'} \cap V$. Two cases are possible.
  1. If $L(x) = \wedge$, by definition of $\text{EX}^{k+1}_{\mathbf{G}'}$ we have that $E'(x) \subseteq \text{EX}^k_{\mathbf{G}'}$. Since $E \subseteq E'$, this implies $E(x) = E(x) \cap V \subseteq E'(x) \cap V \subseteq \text{EX}^k_{\mathbf{G}'} \cap V$ and by induction hypothesis we have $E(x) \subseteq [\![\text{EX}]\!]_{\mathbf{G}}$. By definition of $\Phi^{\text{EX}}_{\mathbf{G}}$, this means $x \in \Phi^{\text{EX}}_{\mathbf{G}}([\![\text{EX}]\!]_{\mathbf{G}}) = [\![\text{EX}]\!]_{\mathbf{G}}$.

2. If $L(x) = \vee$, by definition of $\mathrm{Ex}_{\mathbf{G}'}^{k+1}$ we have that $E'(x) \cap \mathrm{Ex}_{\mathbf{G}'}^{k} \neq \emptyset$. Since $E' = E \cup (E' \setminus E)$, two cases are possible.
   (a) If $E(x) \cap \mathrm{Ex}_{\mathbf{G}'}^{k} \neq \emptyset$, we have $E(x) \cap (\mathrm{Ex}_{\mathbf{G}'}^{k} \cap V) \neq \emptyset$, which by induction hypothesis implies $E(x) \cap [\![\mathrm{Ex}]\!]_{\mathbf{G}} \neq \emptyset$. By definition of $\Phi_{\mathbf{G}}^{\mathrm{Ex}}$, this means $x \in \Phi_{\mathbf{G}}^{\mathrm{Ex}}([\![\mathrm{Ex}]\!]_{\mathbf{G}}) = [\![\mathrm{Ex}]\!]_{\mathbf{G}}$.
   (b) If $(E' \setminus E)(x) \cap \mathrm{Ex}_{\mathbf{G}'}^{k} \neq \emptyset$, and since $G \preceq G'$, there exists $(x, y) \in (E' \setminus E)|_V = (E' \setminus E)|_F$. Thus $x \in F$, which implies by hypothesis $x \in [\![P_\vee \wedge \mathrm{Ex}]\!]_{\mathbf{G}} \subseteq [\![\mathrm{Ex}]\!]_{\mathbf{G}}$.

– To prove $[\![\mathrm{Cx}]\!]_{\mathbf{G}'} \cap V \subseteq [\![\mathrm{Cx}]\!]_{\mathbf{G}}$, we show by induction that $[\![\mathrm{Cx}]\!]_{\mathbf{G}'} \cap V \subseteq \mathrm{Cx}_{\mathbf{G}}^{k}$ for all $k \geq 0$.
  Base step. $[\![\mathrm{Cx}]\!]_{\mathbf{G}'} \cap V \subseteq V = \mathrm{Cx}_{\mathbf{G}}^{0}$.
  Inductive step. Let $x \in [\![\mathrm{Cx}]\!]_{\mathbf{G}'} \cap V = \Phi_{\mathbf{G}'}^{\mathrm{Cx}}([\![\mathrm{Cx}]\!]_{\mathbf{G}'}) \cap V$. Two cases are possible.
  1. If $L(x) = \vee$, by definition of $\Phi_{\mathbf{G}'}^{\mathrm{Cx}}$ we have that $E'(x) \subseteq [\![\mathrm{Cx}]\!]_{\mathbf{G}'}$. Since $E \subseteq E'$, this implies $E(x) = E(x) \cap V \subseteq E'(x) \cap V \subseteq [\![\mathrm{Cx}]\!]_{\mathbf{G}'} \cap V$ and by induction hypothesis we have $E(x) \subseteq \mathrm{Cx}_{\mathbf{G}}^{k}$. By definition of $\Phi_{\mathbf{G}}^{\mathrm{Cx}}$, this means $x \in \Phi_{\mathbf{G}}^{\mathrm{Cx}}(\mathrm{Cx}_{\mathbf{G}}^{k}) = \mathrm{Cx}_{\mathbf{G}}^{k+1}$.
  2. If $L(x) = \wedge$, by definition of $\Phi_{\mathbf{G}'}^{\mathrm{Cx}}$ we have that $E'(x) \cap [\![\mathrm{Cx}]\!]_{\mathbf{G}'} \neq \emptyset$. Since $E' = E \cup (E' \setminus E)$, two cases are possible.
     (a) If $E(x) \cap [\![\mathrm{Cx}]\!]_{\mathbf{G}'} \neq \emptyset$, we have $E(x) \cap ([\![\mathrm{Cx}]\!]_{\mathbf{G}'} \cap V) \neq \emptyset$, which by induction hypothesis implies $E(x) \cap \mathrm{Cx}_{\mathbf{G}}^{k} \neq \emptyset$. By definition of $\Phi_{\mathbf{G}}^{\mathrm{Cx}}$, this means $x \in \Phi_{\mathbf{G}}^{\mathrm{Cx}}(\mathrm{Cx}_{\mathbf{G}}^{k}) = \mathrm{Cx}_{\mathbf{G}}^{k+1}$.
     (b) If $(E' \setminus E)(x) \cap [\![\mathrm{Cx}]\!]_{\mathbf{G}'} \neq \emptyset$, and since $G \preceq G'$, there exists $(x, y) \in (E' \setminus E)|_V = (E' \setminus E)|_F$. Thus $x \in F$, which implies by hypothesis $x \in [\![P_\wedge \wedge \mathrm{Cx}]\!]_{\mathbf{G}} \subseteq [\![\mathrm{Cx}]\!]_{\mathbf{G}} \subseteq \mathrm{Cx}_{\mathbf{G}}^{k+1}$.

Only if. Let $G = (V, E, L, F)$ be a solution-closed EBG. We must show that $F \subseteq [\![(P_\vee \wedge \mathrm{Ex}) \vee (P_\wedge \wedge \mathrm{Cx})]\!]_{\mathbf{G}}$. Suppose there exists $x \in F$ such that $x \notin [\![(P_\vee \wedge \mathrm{Ex}) \vee (P_\wedge \wedge \mathrm{Cx})]\!]_{\mathbf{G}}$. Using the complementarity between EX and CX, this is equivalent to $x \in [\![(P_\vee \wedge \mathrm{Cx}) \vee (P_\wedge \wedge \mathrm{Ex})]\!]_{\mathbf{G}}$. Two cases are possible.

1. If $L(x) = \vee$, then $x \in [\![\mathrm{Cx}]\!]_{\mathbf{G}}$. Let $y \notin V$ and consider the EBG $G' = (V \cup \{y\}, E \cup \{(x, y)\}, L\,[\wedge/y]\,, F)$. It is easy to see that $G \preceq G'$. Since $(E \cup \{(x, y)\})(y) = \emptyset \subseteq [\![\mathrm{Ex}]\!]_{\mathbf{G}'}$, by definition of $\Phi_{\mathbf{G}'}^{\mathrm{Ex}}$ we have $y \in \Phi_{\mathbf{G}'}^{\mathrm{Ex}}([\![\mathrm{Ex}]\!]_{\mathbf{G}'}) = [\![\mathrm{Ex}]\!]_{\mathbf{G}'}$. Because $y \in (E \cup \{(x, y)\})(x)$, this implies $(E \cup \{(x, y)\})(x) \cap [\![\mathrm{Ex}]\!]_{\mathbf{G}'} \neq \emptyset$, which by definition of $\Phi_{\mathbf{G}'}^{\mathrm{Ex}}$ means that $x \in \Phi_{\mathbf{G}'}^{\mathrm{Ex}}([\![\mathrm{Ex}]\!]_{\mathbf{G}'}) = [\![\mathrm{Ex}]\!]_{\mathbf{G}'}$. Thus $x \notin [\![\mathrm{Cx}]\!]_{\mathbf{G}'}$ and $x \in [\![\mathrm{Cx}]\!]_{\mathbf{G}}$ i.e., G is not solution-closed. Contradiction.
2. If $L(x) = \wedge$, then $x \in [\![\mathrm{Ex}]\!]_{\mathbf{G}}$. Let $y \notin V$ and consider the EBG $G' = (V \cup \{y\}, E \cup \{(x, y)\}, L\,[\vee/y]\,, F)$. It is easy to see that $G \preceq G'$. Since $(E \cup \{(x, y)\})(y) = \emptyset \subseteq [\![\mathrm{Cx}]\!]_{\mathbf{G}'}$, by definition of $\Phi_{\mathbf{G}'}^{\mathrm{Cx}}$ we have $y \in \Phi_{\mathbf{G}'}^{\mathrm{Cx}}([\![\mathrm{Cx}]\!]_{\mathbf{G}'}) = [\![\mathrm{Cx}]\!]_{\mathbf{G}'}$. Because $y \in (E \cup \{(x, y)\})(x)$, this implies $(E \cup \{(x, y)\})(x) \cap [\![\mathrm{Cx}]\!]_{\mathbf{G}'} \neq \emptyset$, which by definition of $\Phi_{\mathbf{G}'}^{\mathrm{Cx}}$ means that $x \in \Phi_{\mathbf{G}'}^{\mathrm{Cx}}([\![\mathrm{Cx}]\!]_{\mathbf{G}'}) = [\![\mathrm{Cx}]\!]_{\mathbf{G}'}$. Thus $x \notin [\![\mathrm{Ex}]\!]_{\mathbf{G}'}$ and $x \in [\![\mathrm{Ex}]\!]_{\mathbf{G}}$ i.e., G is not solution-closed. Contradiction.

Proof (Theorem 3). We show the result only for minimal examples, the proof for minimal counterexamples being symmetric.

If. Let $G = (V, E, L, F)$ be an example for $x \in V$ satisfying the conditions a), b), c), d). Let $G' = (V', E', L', F')$ be another example for $x \in V'$ such that $G' \preceq G$. Since by condition a) $\mathbf{G}$ is an Ex-model and by definition G' is a solution-closed subgraph of G, it follows that $\mathbf{G}'$ is also an Ex-model. We prove that $G' = G$.

- To show $V' = V$, suppose that $V \setminus V' \neq \emptyset$. Then there must be $(y, z) \in E$ such that $y \in V'$ and $z \in V \setminus V'$ (because otherwise, by condition c) and since $x \in V'$, $V = E^*(x) \subseteq V' \subseteq V$). By definition of $\preceq$, this means that $y \in F'$, since $(y, z) \in (E \setminus E')|_{V'} = (E \setminus E')|_{F'}$. Two cases are possible.
  1. If $L(y) = \vee$, then $E'(y) = \emptyset$ (i.e., y has no successors in V'), because $1 \leq |(E \setminus E')(y)| \leq |E(y)| = 1$ by condition b). By definition of $\Phi^{\mathrm{Cx}}_{\mathbf{G}'}$, this implies that $y \in \Phi^{\mathrm{Cx}}_{\mathbf{G}'}([\![\mathrm{Cx}]\!]_{\mathbf{G}'}) = [\![\mathrm{Cx}]\!]_{\mathbf{G}'}$ so $\mathbf{G}'$ is not an Ex-model. Contradiction.
  2. If $L(y) = \wedge$, and since $\mathbf{G}'$ is an Ex-model, we have $y \in [\![P_\wedge \wedge \mathrm{Ex}]\!]_{\mathbf{G}'}$ and thus $y \notin [\![(P_\vee \wedge \mathrm{Ex}) \vee (P_\wedge \wedge \mathrm{Cx})]\!]_{\mathbf{G}'}$. Since $y \in F'$, Theorem 2 implies that G' is not solution-closed. Contradiction.
- To show $E' = E$, suppose there exists $(y, z) \in E \setminus E'$. Then, by the same reasoning as above, each of the two possible cases $L(y) = \vee$ and $L(y) = \wedge$ leads to a contradiction.
- To show $L' = L$, we have $L' = L|_{V'} = L|_V = L$ because $V' = V$ and $G' \preceq G$.
- To show $F' = F$, we have $F = F \cap V = F \cap V' \subseteq F'$ because $V' = V$ and $G' \preceq G$. Since G' is solution-closed and $\mathbf{G}'$ is an Ex-model, Theorem 2 implies that $F' \subseteq [\![P_\vee \wedge \mathrm{Ex}]\!]_{\mathbf{G}'} \subseteq [\![P_\vee]\!]_{\mathbf{G}'}$. Finally, by hypothesis and $V' = V$, we have $F' \subseteq [\![P_\vee]\!]_{\mathbf{G}'} = \{y \in V' \mid L(y) = \vee\} = \{y \in V \mid L(y) = \vee\} = F$.

Only if. Let $G = (V, E, L, F)$ be an example for $x \in V$ that is minimal w.r.t. $\preceq$. We must show that G satisfies the conditions a), b), c), d).

Condition a). Suppose that $[\![\mathrm{Cx}]\!]_{\mathbf{G}} \neq \emptyset$. Consider the subgraph $G' = ([\![\mathrm{Ex}]\!]_{\mathbf{G}}, E \cap ([\![\mathrm{Ex}]\!]_{\mathbf{G}} \times [\![\mathrm{Ex}]\!]_{\mathbf{G}}), L|_{[\![\mathrm{Ex}]\!]_{\mathbf{G}}}, [\![P_\vee \wedge \mathrm{Ex}]\!]_{\mathbf{G}})$. G' is strictly smaller than G w.r.t. $\preceq$ because $[\![\mathrm{Ex}]\!]_{\mathbf{G}} \subset V$. Since G is an example for x, this means $x \in [\![\mathrm{Ex}]\!]_{\mathbf{G}}$. We prove that G' is also an example for x. It is sufficient to show that $\mathbf{G}'$ is an Ex-model, because in this case Theorem 2 and the definition of G' imply that G' is also solution-closed. By definition of G' we have $[\![\mathrm{Ex}]\!]_{\mathbf{G}'} \subseteq [\![\mathrm{Ex}]\!]_{\mathbf{G}}$. To prove $[\![\mathrm{Ex}]\!]_{\mathbf{G}} \subseteq [\![\mathrm{Ex}]\!]_{\mathbf{G}'}$, we show by induction that $\mathrm{Ex}^k_{\mathbf{G}} \subseteq \mathrm{Ex}^k_{\mathbf{G}'}$ for all $k \geq 0$.

Base step. $\mathrm{Ex}^0_{\mathbf{G}} = \emptyset \subseteq \mathrm{Ex}^0_{\mathbf{G}'}$.

Inductive step. Let $y \in \mathrm{Ex}^{k+1}_{\mathbf{G}}$. Two cases are possible.

1. If $L(y) = \vee$, by definition of $\mathrm{Ex}^{k+1}_{\mathbf{G}}$ we have that $E(y) \cap \mathrm{Ex}^k_{\mathbf{G}} \neq \emptyset$. By induction hypothesis, this implies $E(y) \cap \mathrm{Ex}^k_{\mathbf{G}'} \neq \emptyset$, which by definitions of $\Phi^{\mathrm{Ex}}_{\mathbf{G}}$ and G' implies $y \in \Phi^{\mathrm{Ex}}_{\mathbf{G}}(\mathrm{Ex}^k_{\mathbf{G}'}) \subseteq \Phi^{\mathrm{Ex}}_{\mathbf{G}}([\![\mathrm{Ex}]\!]_{\mathbf{G}}) = [\![\mathrm{Ex}]\!]_{\mathbf{G}}$. Thus we have $(E \cap ([\![\mathrm{Ex}]\!]_{\mathbf{G}} \times [\![\mathrm{Ex}]\!]_{\mathbf{G}}))(y) \cap \mathrm{Ex}^k_{\mathbf{G}'} \neq \emptyset$. By definition of $\Phi^{\mathrm{Ex}}_{\mathbf{G}'}$, this means $y \in \Phi^{\mathrm{Ex}}_{\mathbf{G}'}(\mathrm{Ex}^k_{\mathbf{G}'}) = \mathrm{Ex}^{k+1}_{\mathbf{G}'}$.

2. If $L(y) = \wedge$, by definition of $\mathrm{Ex}_{\mathbf{G}}^{k+1}$ we have that $E(y) \subseteq \mathrm{Ex}_{\mathbf{G}}^{k}$. By induction hypothesis, this implies $E(y) \subseteq \mathrm{Ex}_{\mathbf{G}'}^{k}$, which by definitions of $\Phi_{\mathbf{G}}^{\mathrm{Ex}}$ and G' implies again $y \in \Phi_{\mathbf{G}}^{\mathrm{Ex}}(\mathrm{Ex}_{\mathbf{G}'}^{k}) \subseteq \Phi_{\mathbf{G}}^{\mathrm{Ex}}([\![\mathrm{Ex}]\!]_{\mathbf{G}}) = [\![\mathrm{Ex}]\!]_{\mathbf{G}}$. Thus we have $(E \cap ([\![\mathrm{Ex}]\!]_{\mathbf{G}} \times [\![\mathrm{Ex}]\!]_{\mathbf{G}}))(y) \subseteq \mathrm{Ex}_{\mathbf{G}'}^{k}$. By definition of $\Phi_{\mathbf{G}'}^{\mathrm{Ex}}$, this means $y \in \Phi_{\mathbf{G}'}^{\mathrm{Ex}}(\mathrm{Ex}_{\mathbf{G}'}^{k}) = \mathrm{Ex}_{\mathbf{G}'}^{k+1}$.

So, G' is an example for x strictly smaller than G. Contradiction.

Condition b). Suppose there exists $y \in V$ such that $L(y) = \vee$ and $|E(y)| \neq 1$. Two cases are possible.

- If $|E(y)| = 0$, then $E(y) = \emptyset \subseteq [\![\mathrm{Cx}]\!]_{\mathbf{G}}$. By definition of $\Phi_{\mathbf{G}}^{\mathrm{Cx}}$, this means $y \in \Phi_{\mathbf{G}}^{\mathrm{Cx}}([\![\mathrm{Cx}]\!]_{\mathbf{G}}) = [\![\mathrm{Cx}]\!]_{\mathbf{G}}$ and thus $\mathbf{G}$ is not an Ex-model. Contradiction.
- If $|E(y)| > 1$, let $u \in E(y)$ such that $K(u) = \min\{K(v) \mid v \in E(y)\}$, where $K(v) = \min\{k \geq 0 \mid v \in \mathrm{Ex}_{\mathbf{G}}^{k}\}$. Intuitively, u is (one of) the first successor(s) of y reached by the increasing chain $\mathrm{Ex}_{\mathbf{G}}^{k}$. Since $|E(y)| > 1$, there exists $z \in E(y) \setminus \{u\}$. Consider the subgraph $G' = (V, E \setminus \{(y,z)\}, L, F \cup \{y\})$. G' is strictly smaller than G w.r.t. $\preceq$ because $E \setminus \{(y,z)\} \subset E$. We prove that G' is also an example for x. Since $\mathbf{G}$ is an Ex-model, it is sufficient to show that $\mathbf{G}'$ is also an Ex-model, because in this case Theorem 2 and the definition of G' imply that G' is also solution-closed. By definition of G' we have $[\![\mathrm{Ex}]\!]_{\mathbf{G}'} \subseteq V$. To prove $V \subseteq [\![\mathrm{Ex}]\!]_{\mathbf{G}'}$, we show by induction that $\mathrm{Ex}_{\mathbf{G}}^{k} \subseteq \mathrm{Ex}_{\mathbf{G}'}^{k}$ for all $k \geq 0$.

  Base step. $\mathrm{Ex}_{\mathbf{G}}^{0} = \emptyset \subseteq \mathrm{Ex}_{\mathbf{G}'}^{0}$.

  Inductive step. Let $v \in \mathrm{Ex}_{\mathbf{G}}^{k+1}$. Two cases are possible.

  1. If $L(v) = \wedge$, by definition of $\mathrm{Ex}_{\mathbf{G}}^{k+1}$ we have that $E(v) \subseteq \mathrm{Ex}_{\mathbf{G}}^{k}$. By definition of G' and because $v \neq y$, we have $(E \setminus \{(y,z)\})(v) = E(v) \subseteq \mathrm{Ex}_{\mathbf{G}}^{k}$, which by induction hypothesis implies $(E \setminus \{(y,z)\})(v) \subseteq \mathrm{Ex}_{\mathbf{G}'}^{k}$. By definition of $\Phi_{\mathbf{G}'}^{\mathrm{Ex}}$, this means $v \in \Phi_{\mathbf{G}'}^{\mathrm{Ex}}(\mathrm{Ex}_{\mathbf{G}'}^{k}) = \mathrm{Ex}_{\mathbf{G}'}^{k+1}$.
  2. If $L(v) = \vee$, by definition of $\mathrm{Ex}_{\mathbf{G}}^{k+1}$ we have that $E(v) \cap \mathrm{Ex}_{\mathbf{G}}^{k} \neq \emptyset$. Two cases are possible.
     (a) If $v \neq y$, by definition of G' we have $(E \setminus \{(y,z)\})(v) = E(v)$, which means that $(E \setminus \{(y,z)\})(v) \cap \mathrm{Ex}_{\mathbf{G}}^{k} \neq \emptyset$. By induction hypothesis, this implies $(E \setminus \{(y,z)\})(v) \cap \mathrm{Ex}_{\mathbf{G}'}^{k} \neq \emptyset$, which by definition of $\Phi_{\mathbf{G}'}^{\mathrm{Ex}}$ means that $v \in \Phi_{\mathbf{G}'}^{\mathrm{Ex}}(\mathrm{Ex}_{\mathbf{G}'}^{k}) = \mathrm{Ex}_{\mathbf{G}'}^{k+1}$.
     (b) If $v = y$, let $w \in E(v) \cap \mathrm{Ex}_{\mathbf{G}}^{k}$. By definition of $K(w)$ and u, this means $k \geq K(w) \geq K(u)$ and thus we have also that $u \in \mathrm{Ex}_{\mathbf{G}}^{k}$. Since $u \neq z$ by definition, we also have that $u \in (E \setminus \{(y,z)\})(v)$ and thus $(E \setminus \{(y,z)\})(v) \cap \mathrm{Ex}_{\mathbf{G}}^{k} \neq \emptyset$. By induction hypothesis, this implies $(E \setminus \{(y,z)\})(v) \cap \mathrm{Ex}_{\mathbf{G}'}^{k} \neq \emptyset$, which by definition of $\Phi_{\mathbf{G}'}^{\mathrm{Ex}}$ means that $v \in \Phi_{\mathbf{G}'}^{\mathrm{Ex}}(\mathrm{Ex}_{\mathbf{G}'}^{k}) = \mathrm{Ex}_{\mathbf{G}'}^{k+1}$.

So, G' is an example for x strictly smaller than G. Contradiction.

Condition c). Suppose that $V \setminus E^*(x) \neq \emptyset$. Consider the subgraph $G' = (E^*(x), E \cap (E^*(x) \times E^*(x)), L|_{E^*(x)}, [\![P_\vee]\!]_{\mathbf{G}} \cap E^*(x))$. G' is strictly smaller than G w.r.t. $\preceq$ because $E^*(x) \subset V$. We prove that G' is also an example for x. It is sufficient to show that $\mathbf{G}'$ is an Ex-model, because in this case Theorem 2 and the definition of G' imply that G' is also solution-closed. By definition of G' we have $[\![\mathrm{Ex}]\!]_{\mathbf{G}'} \subseteq E^*(x)$. Since $\mathbf{G}$ is an Ex-model, we know that $E^*(x) = [\![\mathrm{Ex}]\!]_{\mathbf{G}} \cap E^*(x)$. Thus, in order to prove $E^*(x) \subseteq [\![\mathrm{Ex}]\!]_{\mathbf{G}'}$, we show by induction that $\mathrm{Ex}^k_{\mathbf{G}} \cap E^*(x) \subseteq \mathrm{Ex}^k_{\mathbf{G}'}$ for all $k \geq 0$.

Base step. $\mathrm{Ex}^0_{\mathbf{G}} \cap E^*(x) = \emptyset \subseteq \mathrm{Ex}^0_{\mathbf{G}'}$.

Inductive step. Let $y \in \mathrm{Ex}^{k+1}_{\mathbf{G}} \cap E^*(x)$. Two cases are possible.

1. If $L(y) = \vee$, by definition of $\mathrm{Ex}^{k+1}_{\mathbf{G}}$ we have that $E(y) \cap \mathrm{Ex}^k_{\mathbf{G}} \neq \emptyset$. Since $E(y) \subseteq E^*(x)$, this implies $E(y) \cap (\mathrm{Ex}^k_{\mathbf{G}} \cap E^*(x)) \neq \emptyset$ and by induction hypothesis we have $E(y) \cap \mathrm{Ex}^k_{\mathbf{G}'} \neq \emptyset$. Thus we have $(E \cap (E^*(x) \times E^*(x)))(y) \cap \mathrm{Ex}^k_{\mathbf{G}'} \neq \emptyset$. By definition of $\Phi^{\mathrm{Ex}}_{\mathbf{G}'}$, this means $y \in \Phi^{\mathrm{Ex}}_{\mathbf{G}'}(\mathrm{Ex}^k_{\mathbf{G}'}) = \mathrm{Ex}^{k+1}_{\mathbf{G}'}$.
2. If $L(y) = \wedge$, by definition of $\mathrm{Ex}^{k+1}_{\mathbf{G}}$ we have that $E(y) \subseteq \mathrm{Ex}^k_{\mathbf{G}}$. Since $E(y) \subseteq E^*(x)$, this implies $E(y) = E(y) \cap E^*(x) \subseteq \mathrm{Ex}^k_{\mathbf{G}} \cap E^*(x)$ and by induction hypothesis we have $E(y) \subseteq \mathrm{Ex}^k_{\mathbf{G}'}$. Thus we have $(E \cap (E^*(x) \times E^*(x)))(y) \subseteq \mathrm{Ex}^k_{\mathbf{G}'}$. By definition of $\Phi^{\mathrm{Ex}}_{\mathbf{G}'}$, this means $y \in \Phi^{\mathrm{Ex}}_{\mathbf{G}'}(\mathrm{Ex}^k_{\mathbf{G}'}) = \mathrm{Ex}^{k+1}_{\mathbf{G}'}$.

So, G' is an example for x strictly smaller than G. Contradiction.

Condition d). Since G is solution-closed and $\mathbf{G}$ is an Ex-model, by Theorem 2 we have that $F \subseteq [\![P_\vee \wedge \mathrm{Ex}]\!]_{\mathbf{G}} \subseteq [\![P_\vee]\!]_{\mathbf{G}}$. To show $[\![P_\vee]\!]_{\mathbf{G}} \subseteq F$, suppose there exists $y \in V$ such that $L(y) = \vee$ and $y \notin F$. Consider the subgraph $G' = (V, E, L, F \cup \{y\})$. G' is strictly smaller than G w.r.t. $\preceq$ because $G' \neq G$. We prove that G' is also an example for x. Since $\mathbf{G}$ is an Ex-model and $\mathbf{G}' = \mathbf{G}$ by definition of G', it follows that $\mathbf{G}'$ is also an Ex-model. Because $F \cup \{y\} \subseteq [\![P_\vee]\!]_{\mathbf{G}} = \{z \in V \mid L(z) = \vee\} = [\![P_\vee]\!]_{\mathbf{G}'}$, Theorem 2 implies that G' is also solution-closed. So, G' is an example for x strictly smaller than G. Contradiction.

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