Efficient Diagnostic Generation for Boolean Equation Systems

Radu Mateescu

Inria Rhône-Alpes / Vasy
655, avenue de l'Europe, F-38330 Montbonnot Saint Martin, France
Radu.Mateescu@inria.fr

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.

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

S. Graf and M. Schwartzbach (Eds.): TACAS/ETAPS 2000, LNCS 1785, pp. 251–265, 2000.

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 [7,6], 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 (BES):
$M = \{x_i \overset{\sigma}{=} op_i X_i\}_{1 \leq i \leq n}$
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 logic we use (see Figure 2) is a variant of the alternation-free 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 $\rho : \mathcal{Y} \to 2^V$:
$[\![P_\vee]\!]_{\mathbf{G}}\rho = \{x \in V \mid L(x) = \vee\}$
$[\![P_\wedge]\!]_{\mathbf{G}}\rho = \{x \in V \mid L(x) = \wedge\}$
$[\![\varphi_1 \vee \varphi_2]\!]_{\mathbf{G}}\rho = [\![\varphi_1]\!]_{\mathbf{G}}\rho \cup [\![\varphi_2]\!]_{\mathbf{G}}\rho$
$[\![\varphi_1 \wedge \varphi_2]\!]_{\mathbf{G}}\rho = [\![\varphi_1]\!]_{\mathbf{G}}\rho \cap [\![\varphi_2]\!]_{\mathbf{G}}\rho$
$[\![\langle - \rangle\,\varphi]\!]_{\mathbf{G}}\rho = \{x \in V \mid E(x) \cap [\![\varphi]\!]_{\mathbf{G}}\rho \neq \emptyset\}$
$[\![[-]\,\varphi]\!]_{\mathbf{G}}\rho = \{x \in V \mid E(x) \subseteq [\![\varphi]\!]_{\mathbf{G}}\rho\}$
$[\![Y]\!]_{\mathbf{G}}\rho = \rho(Y)$
$[\![\mu Y.\varphi]\!]_{\mathbf{G}}\rho = \bigcap\{U \subseteq V \mid \Phi_{\mathbf{G}\rho}(U) \subseteq U\}$
$[\![\nu Y.\varphi]\!]_{\mathbf{G}}\rho = \bigcup\{U \subseteq V \mid U \subseteq \Phi_{\mathbf{G}\rho}(U)\}$
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:*

$$\text{Ex} = \mu Y.(P_\vee \wedge \langle - \rangle Y) \vee (P_\wedge \wedge [-] Y)$$
$$\text{Cx} = \nu Y.(P_\vee \wedge [-] Y) \vee (P_\wedge \wedge \langle - \rangle Y)$$

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

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

Theorem 1 (characterization of BES solution). *Let $M = \{x_i \stackrel{\mu}{=} op_i X_i\}_{1 \leq i \leq n}$ 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

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

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.

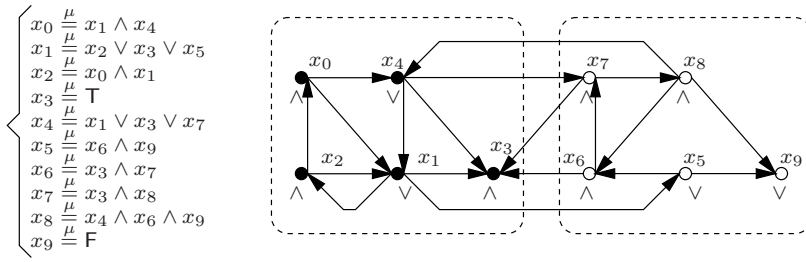

Fig. 3. A closed BES and its associated EBG

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 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 $x \models_{\mathbf{G_1}} \text{EX}$ iff $x \models_{\mathbf{G_2}} \text{EX}$. 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* EBGs. 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* $G_1 \preceq G_2$:

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

or, equivalently:

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

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* $x \models_{\mathbf{G}} \text{Ex}$ *and* counterexample *if* $x \models_{\mathbf{G}} \text{Cx}$.

The following theorem 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:*

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

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 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 ($\Phi^{\text{Ex}}_{\mathbf{G}}$ 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.

$\mathbf{G} = (V, E, L)$
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** $\quad \mathbf{I_1} \wedge \mathbf{I_2} \wedge \mathbf{I_3}$
    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

$\{x \in V \mid c(x) = 0\} = [\![\mathrm{Ex}]\!]_{\mathbf{G}} \wedge$
$\{(x, y) \in E \mid x, y \in [\![\mathrm{Ex}]\!]_{\mathbf{G}} \wedge (L(x) = \vee \Rightarrow y = s(x))\}$ is acyclic

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

Fig. 4. Extended global resolution algorithm

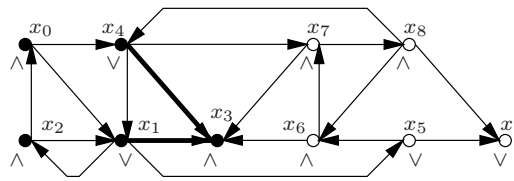

Fig. 5. Computation of c and s by SOLVE

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.

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 $x \in [\![\text{Ex}]\!]_{\mathbf{G}}$, and for every $\vee$-vertex $y \in [\![\text{Ex}]\!]_{\mathbf{G}}$ a successor $s(y)$ as computed by the SOLVE algorithm given in Section 4.1.

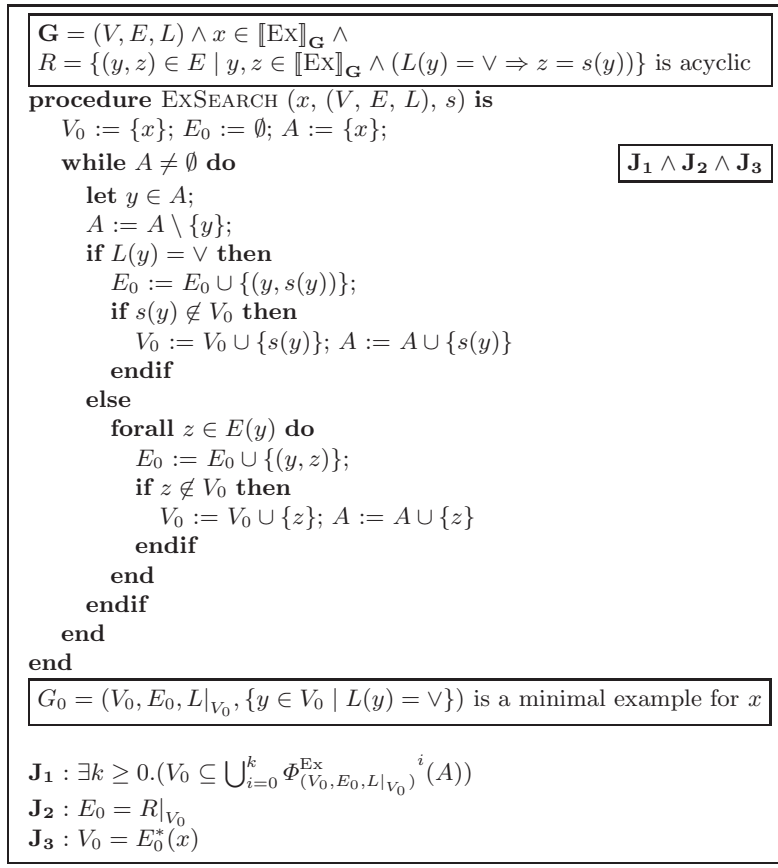

Fig. 6. Minimal example generation algorithm

ExSearch iteratively accumulates in V_0 all the vertices in $[\![\text{Ex}]\!]_{\mathbf{G}}$ 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 $V_0 \subseteq \bigcup_{i \geq 0} \Phi^{\text{Ex}\ i}_{\mathbf{G_0}}(\emptyset) = \mu\Phi^{\text{Ex}}_{\mathbf{G_0}} = [\![\text{Ex}]\!]_{\mathbf{G_0}} \subseteq V_0$. 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.

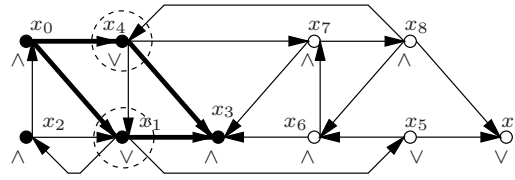

Fig. 7. A minimal example for x_0 computed by ExSearch

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 $x_0 \models_{\mathbf{G}_0} \text{Cx}$. 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.

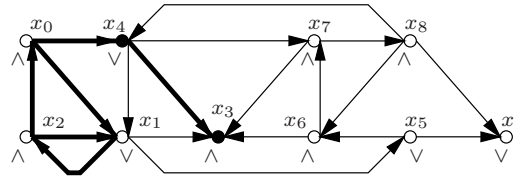

Fig. 8. An erroneous example for x_0 computed in absence of s

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 $\mathsf{E}[\mathsf{T}\,\mathsf{U}\,\varphi]$).

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.

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

procedure CxSearch $(x, (V, E, L), c)$ **is**
  $V_0 := \{x\}$; $E_0 := \emptyset$; $A := \{x\}$;
  while $A \neq \emptyset$ **do** $\qquad \boxed{\mathbf{K_1} \wedge \mathbf{K_2} \wedge \mathbf{K_3}}$
    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

CxSearch iteratively accumulates in V_0 all the vertices in $[\![\text{Cx}]\!]_{\mathbf{G}}$ 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 $\mathbf{K_1}$ ($\Phi^{\text{Cx}}_{\mathbf{G}}$ 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 $V_0 \subseteq \Phi^{\text{Cx}}_{\mathbf{G_0}}(V_0)$. By Tarski's theorem [27], this implies $V_0 \subseteq \nu\Phi^{\text{Cx}}_{\mathbf{G_0}} = [\![\text{Cx}]\!]_{\mathbf{G_0}} \subseteq V_0$. 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.

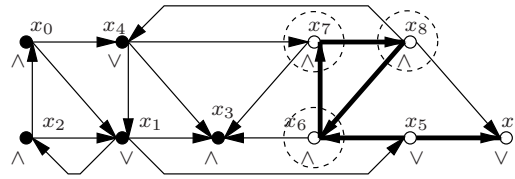

Fig. 10. A minimal counterexample for x_5 computed by CxSearch

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 $[\![\text{Cx}]\!]_{\mathbf{G}}$ (this happens for Ctl formulas like $\mathsf{A}[\mathsf{T}\,\mathsf{U}\,\varphi]$).

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 non-trivial 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. *TCS*, 126(1):3–30, 1994. 251, 254, 257, 259, 263
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 CAV'95 (Liege, Belgium)*, vol. 939 of LNCS, pp. 142–154. Springer Verlag, July 1995. 251, 263
3. A. Arnold and P. Crubillé. A Linear Algorithm to Solve Fixed-point Equations on Transition Systems. *Information Processing Letters*, 29:57–66, 1988. 251, 257, 259, 263
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. 252
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. 251, 252, 263
6. E. M. Clarke, O. Grumberg, K. L. McMillan, and X. Zhao. Efficient Generation of Counterexamples and Witnesses in Symbolic Model Checking. In *Proceedings of DAC'95 (San Francisco, CA, USA)*, pp. 427–432. ACM, June 1995. 252
7. 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 (Noordwijkerhout, The Netherlands)*, vol. 803 of LNCS, pp. 124–175. Springer Verlag, June 1993. 252
8. R. Cleaveland. On Automatically Explaining Bisimulation Inequivalence. In E. M. Clarke and R. P. Kurshan, editors, *Proceedings of CAV'90 (New Brunswick, NJ, USA)*, vol. 531 of LNCS, pp. 364–372. Springer Verlag, June 1990. 252
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 CAV'91 (Aalborg, Denmark)*, vol. 575 of LNCS, pp. 48–58. Springer Verlag, July 1991. 251, 252, 254, 257, 259, 263
10. E. A. Emerson and C-L. Lei. Efficient Model Checking in Fragments of the Propositional Mu-Calculus. In *Proceedings of the 1st LICS*, pp. 267–278, 1986. 253
11. J-C. Fernandez, H. Garavel, A. Kerbrat, R. Mateescu, L. Mounier, and M .Sighireanu. CADP (CÆSAR/ALDEBARAN Development Package): A Protocol Validation and Verification Toolbox. In R. Alur and T. A. Henzinger, editors, *Proceedings of CAV'96 (New Brunswick, NJ, USA)*, vol. 1102 of LNCS, pp. 437–440. Springer Verlag, August 1996. 263
12. J-C. Fernandez, C. Jard, T. Jéron, L. Nedelka, and C. Viho. Using On-the-Fly Verification Techniques for the Generation of Test Suites. In R. Alur and T. A. Henzinger, editors, *Proceedings of CAV'96 (New Brunswick, NJ, USA)*, vol. 1102 of LNCS, pp. 348–359. Springer Verlag, August 1996. 263

13. J-C. Fernandez and L. Mounier. "On the Fly" Verification of Behavioural Equivalences and Preorders. In K. G. Larsen and A. Skou, editors, *Proceedings of CAV'91 (Aalborg, Denmark)*, vol. 575 of LNCS. Springer Verlag, July 1991. 252
14. H. Garavel. OPEN/CÆSAR: An Open Software Architecture for Verification, Simulation, and Testing. In B. Steffen, editor, *Proceedings of TACAS'98 (Lisbon, Portugal)*, vol. 1384 of LNCS, pp. 68–84. Springer Verlag, March 1998. 263
15. H. Korver. Computing Distinguishing Formulas for Branching Bisimulation. In K. G. Larsen and A. Skou, editors, *Proceedings of CAV'91 (Aalborg, Denmark)*, vol. 575 of LNCS, pp. 13–23. Springer Verlag, July 1991. 252
16. D. Kozen. Results on the Propositional μ-calculus. *TCS*, 27:333–354, 1983. 252
17. K. G. Larsen. Efficient Local Correctness Checking. In G. v. Bochmann and D. K. Probst, editors, *Proceedings of CAV'92 (Montréal, Canada)*, vol. 663 of LNCS, pp. 30–43. Springer Verlag, June-July 1992. 251
18. X. Liu, C. R. Ramakrishnan, and S. A. Smolka. Fully Local and Efficient Evaluation of Alternating Fixed Points. In B. Steffen, editor, *Proceedings of TACAS'98 (Lisbon, Portugal)*, vol. 1384 of LNCS. Springer Verlag, March 1998. 251, 263
19. X. Liu and S. A. Smolka. Simple Linear-Time Algorithms for Minimal Fixed Points. In K. G. Larsen, S. Skyum, and G. Winskel, editors, *Proceedings of ICALP'98 (Aalborg, Denmark)*, vol. 1443 of LNCS. Springer Verlag, July 1998. 251, 259, 263
20. A. Mader. *Verification of Modal Properties Using Boolean Equation Systems*. VERSAL 8, Bertz Verlag, Berlin, 1997. 251, 254, 263
21. K. L. McMillan. *Symbolic Model Checking*. Kluwer Academic Publishers, 1993. 252
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*, vol. 469 of LNCS, pp. 407–419. Springer Verlag, 1990. 263
23. A. Rasse. Error Diagnosis in Finite Communicating Systems. In K. G. Larsen and A. Skou, editors, *Proceedings of CAV'91 (Aalborg, Denmark)*, vol. 575 of LNCS, pp. 114–124. Springer Verlag, July 1991. 252
24. P. Stevens and C. Stirling. Practical Model-Checking using Games. In B. Steffen, editor, *Proceedings of TACAS'98 (Lisbon, Portugal)*, vol. 1384 of LNCS, pp. 85–101. Springer Verlag, March 1998. 252
25. C. Stirling. Bisimulation, model checking and other games. In Notes for Mathfit instructional meeting on games and computation, Edinburgh, June 1997. 252
26. R. Streett. Propositional Dynamic Logic of Looping and Converse. *Information and Control*, (54):121–141, 1982. 263
27. A. Tarski. A Lattice-Theoretical Fixpoint Theorem and its Applications. *Pacific Journal of Mathematics*, (5):285–309, 1955. 262
28. B. Vergauwen and J. Lewi. A Linear Algorithm for Solving Fixed-point Equations on Transition Systems. In *Proceedings of CAAP'92 (Rennes, France)*, vol. 581 of LNCS, pp. 322–341. Springer Verlag, February 1992. 251, 257, 259, 263
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 ICALP'94 (Vienna)*, vol. 820 of LNCS, pp. 304–315. Springer Verlag, July 1994. 251, 259, 263
30. B. Vergauwen, J. Wauman, and J. Lewi. Efficient FixPoint Computation. In *Proceedings of SAS'94 (Namur, Belgium)*, vol. 864 of LNCS, pp. 314–328. Springer Verlag, September 1994. 251, 257, 259