The Modal μ-Calculus, Model Checking, Equation Systems and Gauß Elimination

Angelika Mader*
Technische Universität München, Germany

1 Introduction

The modal μ-calculus [Koz83, Sti92] is a powerful logic. It is particularly useful for expressing properties of parallel processes with finite (or even infinite) state spaces; it finds application in process algebra [Wal89] and in Petri nets [Bra92]. Proving whether a property expressed in the modal μ-calculus holds for particular states of a process is called model checking [CE81, CES86]. Various algorithms are available. The main approaches are model checkers based on the fixpoint approximation [EmL86, CDS92, And92, BCMDH92, LBCJM94] and tableau based model checkers [StW89, Cle90, Lar92, Mad92]. One important technique consists of the transformation of a property and a model to a (Boolean) equation system [AC88, And92, CDS92, Lar92, VeL92]. Then model checking is equivalent to the computation of a certain fixpoint. In fact, various correctness problems may be represented in this way.

In this paper we present a novel, algebraic approach for solving Boolean equation systems. It does not use approximation techniques and therefore does not require backtracking. The method works straightforward by successively eliminating variables and reducing the Boolean equation system, similar to Gauß elimination for linear equation systems. Homogeneous, hierarchical and alternating fixpoints are treated uniformly. Contrary to other techniques Gauß elimination leads to both a global and a local model checking algorithm within one framework.

The elimination of a variable is based on a simple observation: the equation $X = A(X)$ (with monotone A) has the least fixpoint $A(\text{false})$ and the greatest fixpoint $A(\text{true})$. The reduction of a Boolean equation system is done by syntactical substitution of variables by expressions.

The difference between the global version of Gauß elimination and the local one can be characterized as follows: The global version solves the whole equation system, whereas the local version only takes a subset of equations into account which is necessary to determine the variable of interest. The selection of a suitable subset of equations is demand-driven. Whereas the global version is more of theoretical interest (approximation

*Institut für Informatik, Technische Universität München, Arcisstr.21, D-80333 München, Germany, email: mader@informatik.tu-muenchen.de. Supported by Siemens AG, Corporate Research and Development, and a grant from the Hochschulsonderprogramm II.

techniques have better worst case complexity), the local version has advantages in the context of model checking. It is closely related to the tableau methods, and can be interpreted as a combination of top-down strategy of the tableau method and bottom-up evaluation which avoids redundancy caused by recomputation of subtableaux. Therefore it is much more efficient than the tableau methods and has a better worst case complexity.
Section 2 introduces Boolean equation systems and their solution. Gauß elimination for Boolean equation systems is presented in section 3. Section 4 contains a short introduction into the modal μ-calculus, and the transformation of the model checking problem into a equation solving problem. Comparison with other work is discussed in section 5. Examples are in section 6. Section 7 is the conclusion. The appendix contains correctness proofs.

2 Boolean Equation Systems

In this section we define Boolean equation systems and what we regard as solution of a Boolean equation system.

Definition 1 Let $\mathcal{X} = \{X_1, \ldots, X_n\}$ be a set of Boolean variables, $<$ a linear order on $\mathcal{X}$, and $\{A_1, \ldots, A_n\}$ a set of negation free Boolean expressions containing variables from $\mathcal{X}$. Then the set of labelled equations $E_i : X_i \stackrel{\sigma_i}{=} A_i$, where $\sigma_i \in \{\mu, \nu\}$, is a Boolean equation system.

In the following we assume that the order on the variables is according to their indices.
As the Boolean expressions are negation free and therefore monotone the equation system (the plain one without order and labels) has a set of fixpoints. In the context here we are interested in a distinguished fixpoint which we call the solution of the Boolean equation system. Below we give the definition of the solution.
We introduce some notation first. The vector $(X_1, \ldots, X_n)$ of Boolean variables will be abbreviated by $\underline{X}$; analogously $\underline{\sigma}$, $\underline{A}$ and $\underline{E}$ denote the vectors of labels, expressions and equations respectively. A Boolean equation system can now be written as: $\underline{E}$: $\underline{X} \stackrel{\underline{\sigma}}{=} \underline{A}(\underline{X})$.
Further abbreviations will be used. $\underline{Y}^{(i)}$ stands for the i-th rest $(Y_i, \ldots Y_n)$ of the Boolean vector $\underline{Y}$, and again analogously $\underline{\sigma}^{(i)}$, $\underline{A}^{(i)}$, and $\underline{E}^{(i)}$ denote the i-the rests of the related vectors. By $\underline{E}^{(i)}[Y_j/X_j]$ we mean the equation system $\underline{E}^{(i)}$ where all unbound occurrences of X_j on the right-hand-side of the equations are substituted by Y_j.
Now for $\underline{\sigma} \in \{\mu, \nu\}^n$ we define the **lexicographic order** $<_{\underline{\sigma}}$ on the Boolean vector space $\mathbb{B}^n$. Let $<_\sigma$ on $\mathbb{B}$ for $\sigma \in \{\mu, \nu\}$ be such that $\mathsf{false} <_\mu \mathsf{true}$ and $\mathsf{true} <_\nu \mathsf{false}$.

Definition 2 Let $\underline{Y}, \underline{Y}' \in \mathbb{B}^n$, $\underline{\sigma} \in \{\mu, \nu\}^n$. The lexicographic order on $\underline{Y}, \underline{Y}'$ is defined as $\underline{Y} <_{\underline{\sigma}} \underline{Y}' :\Leftrightarrow \quad \exists i, 1 \le i \le n : \; Y_i <_{\sigma_i} Y_i' \;$ and $\; \forall j, 1 \le j < i : \; Y_j = Y_j'$.

Definition 3 $\underline{Y}^{(i)} \in \mathbb{B}^{n-i+1}$ is the solution of the equation system $\underline{E}^{(i)}[Y_1/X_1, \ldots Y_{i-1}/X_{i-1}]$

$:\Leftrightarrow$ if $i = n$, then $\underline{Y}^{(i)}$ is the least fixpoint of $\underline{E}^{(i)}[Y_1/X_1, \ldots Y_{i-1}/X_{i-1}]$ wrt. $<_{\sigma_n}$,

if $i < n$, then $\underline{Y}^{(i)}$ is the least one wrt. $<_{\underline{\sigma}^{(i)}}$ of those fixpoints of $\underline{E}^{(i)}[Y_1/X_1, \ldots Y_{i-1}/X_{i-1}]$ which satisfy the following property:

$\underline{Y}^{(i+1)}$ is solution of $\underline{E}^{(i+1)}[Y_1/X_1, \ldots Y_{i-1}/X_{i-1}, Y_i/X_i]$.

There exist several algorithms to determine the set of fixpoints of a Boolean equation system; for examples see [Rud74]. However, even if the set of fixpoints is given, it is not trivial to select the one fixpoint which satisfies the definition of the solution above. This indicates that the existing equation solving methods do not help in our case. We will illustrate this by two small examples.

The first example shows that the solution is not the lexicographic least fixpoint. The equation system $X_2 \stackrel{\mu}{=} X_2$ has two fixpoints true and false. With respect to the order $<_\mu$ false is the least one. Now consider the equation system $X_1 \stackrel{\nu}{=} X_2, X_2 \stackrel{\mu}{=} X_2$, where $X_1 < X_2$. The lexicographic least fixpoint is $(\mathsf{true}, \mathsf{true})$, whereas $(\mathsf{false}, \mathsf{false})$ is the solution as indicated by the first equation system and as defined above.

In the following example two Boolean equation systems are given, both having the same set of fixpoints and the same labels on the equations, but different solutions. The equation system $X_1 \stackrel{\nu}{=} X_2, X_2 \stackrel{\mu}{=} X_2$, where $X_1 < X_2$, has the fixpoints $(\mathsf{true}, \mathsf{true})$ and $(\mathsf{false}, \mathsf{false})$. The solution is $(\mathsf{false}, \mathsf{false})$ as in the previous example.

The equation system $X_1 \stackrel{\nu}{=} X_2, X_2 \stackrel{\mu}{=} X_1$, where $X_1 < X_2$, also has the fixpoints $(\mathsf{true}, \mathsf{true})$ and $(\mathsf{false}, \mathsf{false})$, but the solution here is $(\mathsf{true}, \mathsf{true})$.

3 Gauß Elimination

In this section we present two algorithms which determine the solution of a Boolean equation system as in definition 3. In contrast to other methods we do not make use of approximation and backtracking techniques. Instead we stepwise reduce a Boolean equation system to a Boolean equation systemconsisting one equation and one variable less. The steps of eliminating a variable from an expression and of substituting variables by expressions remind very much to Gauß elimination for linear equation systems.

The following propositions are the basis for the Gauß elimination. Proofs are contained in the appendix.

Proposition 1 For the solution Y of the equation system $X \stackrel{\sigma}{=} A(X)$ consisting of one single equation it holds:

$$Y = \begin{cases} A(\mathsf{false}) & \text{if } \sigma = \mu \\ A(\mathsf{true}) & \text{if } \sigma = \nu \end{cases}$$

Proposition 1 can be extended to expressions and equation systems. It allows a representation of the Boolean expression A_i with no occurrence of X_i. In the algorithm we will call this the Gauß division step.

Proposition 2 $\underline{Y}$ is the solution of the Boolean equation system $\underline{E}$ of the form $\underline{X} \stackrel{\underline{\sigma}}{=} \underline{A}(\underline{X})$, iff $\underline{Y}$ also is the solution of the modified Boolean equation system $\underline{F}$, where the equations are of the following form:

$$\begin{array}{lcl} X_1 & \stackrel{\sigma_1}{=} & A_1(\mathsf{b}_1, X_2, \qquad \ldots \qquad, X_n) \\ \vdots & & \\ X_i & \stackrel{\sigma_i}{=} & A_i(X_1, \ldots, X_{i-1}, \mathsf{b}_i, X_{i+1}, \ldots, X_n) \\ \vdots & & \\ X_n & \stackrel{\sigma_n}{=} & A_n(X_1, \qquad \ldots \qquad, X_{n-1}, \mathsf{b}_n) \end{array} \qquad \text{where} \quad \mathsf{b}_i = \begin{cases} \mathsf{true} & \text{if } \sigma_i = \nu \\ \mathsf{false} & \text{if } \sigma_i = \mu \end{cases}$$

The next proposition shows that every occurrence of a variable X_n may be substituted by the expression A_n, in which X_n has been eliminated. We call this the Gauß elimination step.

Proposition 3 The Boolean vector $\underline{Y}$ is the solution of the equation system $\underline{E}$, iff it is the solution of the equation system $\underline{G}$, where $\underline{G}$ is the modified equation system:

$$
\begin{array}{rcl}
X_1 & \overset{\sigma_1}{=} & A_1 \quad (X_1, \ldots, X_{n-1}, A_n(X_1, \ldots, X_{n-1}, \mathsf{b}_n)) \\
\vdots & & \\
X_{n-1} & \overset{\sigma_{n-1}}{=} & A_{n-1}(X_1, \ldots, X_{n-1}, A_n(X_1, \ldots, X_{n-1}, \mathsf{b}_n)) \\
X_n & \overset{\sigma_n}{=} & A_n \quad (X_1, \ldots, X_{n-1}, \mathsf{b}_n)
\end{array}
$$

Based on these two Gauß steps we now propose two algorithms to determine the solution of a Boolean equation system. One algorithm operates on the whole equation system; this is the global version of Gauß elimination. The basic idea is that a Boolean equation system can be reduced to a Boolean equation system with the same solution, but one equation less. The reduction is performed by an elimination step, where in the last equation, say $X_j \overset{\sigma_j}{=} A_j(X_1, \ldots, X_j)$, all occurrences of X_j on the right hand side are instantiated by $\mathsf{b}_j = \mathsf{true}$ or $\mathsf{b}_j = \mathsf{false}$ depending on σ_j, and a substitution step, where in all other equations each occurrence of X_j is substituted by the expression $A_j(X_1, \ldots, X_{j-1}, \mathsf{b}_j)$. The result is an equation system with no free occurrence of X_j. Now the same reduction can be applied to the equation system consisting of the first $j-1$ equations and so on. In the end we get a variable free expression for the variable X_1.

Assume $\underline{X} \overset{\sigma}{=} \underline{A}(\underline{X})$ as input;
i := n;
while not ($A_1 = \mathsf{true}$ **or** $A_1 = \mathsf{false}$)
do
Instantiate X_i in A_i to $\{\mathsf{true}, \mathsf{false}\}$; (Gauß-division)
Substitute A_i for X_i in $A_1, \ldots, A_{i-1}$; (Elimination step)
A_1:=Eval(A_1); ... ; A_{i-1}:=Eval(A_{i-1}) (Evaluation step)
i := i - 1;
od

Figure 1: Global Version of Gauß Elimination

In most contexts we are only interested in the first component of the solution, i.e. whether X_1 is true or false. Therefore the algorithm in figure 1 stops, if the solution of X_1 (A_1) is determined. If we are interested in the whole solution the Gauß division step and elimination step have to be applied n times giving an expression for every X_i where the variables $X_i, \ldots, X_n$ do not occur. A straight backward substitution leads to the whole solution.

If only the first variable is of interest, it suffices to consider only the subset of equations which is necessary to determine the solution for X_1. The relevant subset of equations is selected in a top-down manner. This observation leads to the local version of Gauß elimination given in figure 2. The idea is as follows. We start with the equation system $\underline{E}'$ consisting only of the equation $X_1 \stackrel{\sigma_1}{=} A_1(X_1, \ldots, X_j)$. As long as X_1 is not evaluated to true or false we select a free variable from A_1, insert its equation in $\underline{E}'$, apply the global version of Gauß elimination, and continue in the same way with the modified equation system $\underline{E}'$.

Create E_1 and let E_1 be $\underline{E}'$;
Instantiate X_1 in A_1; (Gauß-division)
$A_1 := \mathrm{Eval}(A_1)$; (Evaluation step)
while not ($A_1 = \mathsf{true}$ **or** $A_1 = \mathsf{false}$)
do
Select X_j, where j is such that $E_j \notin \underline{E}'$;
Create E_j, insert E_j in $\underline{E}'$ and extend the order on $\underline{E}'$ to E_j;
Apply Gauß-elimination on $\underline{E}'$
od

Figure 2: Local Model Checking Algorithm

4 The Modal μ-Calculus and Model Checking

This section gives a brief introduction to the modal μ-calculus. For details see [Sti92].

The syntax of the modal μ-calculus is defined with respect to a set $\mathcal{Q}$ of atomic propositions including true and false, a denumerable set $\mathcal{Z}$ of propositional variables and a finite set $\mathcal{L}$ of action labels. The set μM of modal μ-calculus assertions is determined by the following grammar:

$\Phi ::= Z \mid Q \mid \Phi \wedge \Phi \mid \Phi \vee \Phi \mid [a]\Phi \mid \langle a \rangle \Phi \mid \mu Z.\Phi \mid \nu Z.\Phi$

M denotes the set of variable and fixpoint free assertions, i.e., the expressions of the **propositional modal logic**, Π_0 denotes the set of fixpoint free assertions, $M \subset \Pi_0$. In the following an expression of the form $\sigma Z.\Phi$, where $\sigma \in \{\mu, \nu\}$, is called a **fixpoint expression**. Formulae of the modal μ-calculus with the set $\mathcal{L}$ of action labels are interpreted relative to a **labelled transition system** $\mathcal{T} = (\mathcal{S}, \{\stackrel{a}{\rightarrow} \mid a \in \mathcal{L}\})$, where $\mathcal{S}$ is a finite set of states and $\stackrel{a}{\rightarrow} \subseteq \mathcal{S} \times \mathcal{S}$ for every $a \in \mathcal{L}$ is a binary relation on states. A **valuation function** $\mathcal{V}$ assigns to every propositional variable Z and atomic proposition Q a set of states $\mathcal{V}(Z) \subseteq \mathcal{S}$ and $\mathcal{V}(Q) \subseteq \mathcal{S}$. Let V[S'/Z] be the valuation such that V[S'/Z](Z) = S', and otherwise as V. The pair $\mathcal{T}$ and $\mathcal{V}$ is called a **model** of the μ-calculus. The semantics of each μ-calculus formula Φ is the set of states $\|\Phi\|^{\mathcal{T}}_{\mathcal{V}}$ defined inductively as follows:

$$\begin{array}{rcl@{\qquad}rcl}
\|Z\|^{\mathcal{T}}_{\mathcal{V}} & = & \mathcal{V}(Z) & \|\Phi_1 \vee \Phi_2\|^{\mathcal{T}}_{\mathcal{V}} & = & \|\Phi_1\|^{\mathcal{T}}_{\mathcal{V}} \cup \|\Phi_2\|^{\mathcal{T}}_{\mathcal{V}} \\
\|Q\|^{\mathcal{T}}_{\mathcal{V}} & = & \mathcal{V}(Q) & \|\Phi_1 \wedge \Phi_2\|^{\mathcal{T}}_{\mathcal{V}} & = & \|\Phi_1\|^{\mathcal{T}}_{\mathcal{V}} \cap \|\Phi_2\|^{\mathcal{T}}_{\mathcal{V}}
\end{array}$$

$$\begin{array}{rcl}
\|[a]\Phi\|^{\mathcal{T}}_{\mathcal{V}} & = & [\![a]\!]^{\mathcal{T}}\ \|\Phi\|^{\mathcal{T}}_{\mathcal{V}}, \quad \text{where } [\![a]\!]^{\mathcal{T}} S' = \{s \mid \forall s' \in S.\ \text{if } s \xrightarrow{a} s' \text{ then } s' \in S'\} \\
\|\langle a\rangle\Phi\|^{\mathcal{T}}_{\mathcal{V}} & = & \langle\!\langle a\rangle\!\rangle^{\mathcal{T}}\ \|\Phi\|^{\mathcal{T}}_{\mathcal{V}}, \quad \text{where } \langle\!\langle a\rangle\!\rangle^{\mathcal{T}} S' = \{s \mid \exists s' \in S'.s \xrightarrow{a} s'\} \\
\|\mu Z.\Phi\|^{\mathcal{T}}_{\mathcal{V}} & = & \bigcap\{S' \subseteq \mathcal{S} \mid \|\Phi\|^{\mathcal{T}}_{\mathcal{V}[S'/Z]} \subseteq S'\} \\
\|\nu Z.\Phi\|^{\mathcal{T}}_{\mathcal{V}} & = & \bigcup\{S' \subseteq \mathcal{S} \mid S' \subseteq \|\Phi\|^{\mathcal{T}}_{\mathcal{V}[S'/Z]}\}
\end{array}$$

Given a model $\mathcal{M} = (\mathcal{T}, \mathcal{V})$ model checking is to examine the question whether a certain expression Φ holds for the initial state $s \in \mathcal{S}$ of the transition system $\mathcal{T}$, i.e., whether $s \in \|\Phi\|^{\mathcal{T}}_{\mathcal{V}}$. We transform the model checking problem into the problem of solving a Boolean equation system. This was already done by several authors, e.g. see [AC88, And92, Lar92, ClS91, CDS92, VeL92]. In contrary to their approaches here arbitrary negation free expressions are considered as right hand sides of the equations (no restriction to simple expressions). Furthermore we create one Boolean equation system for the whole problem with a partial order defined on its variables and equations.

Roughly the transformation is performed by the following steps: a fixpoint expression can be represented by an equation system with an additional ordering on the equations. On the semantic part we interpret the equation system with respect to a model, i.e., a fixpoint equation of modal logic becomes a fixpoint equation over the powerset of a state space. An isomorphic representation of a powerset of states is a Boolean vector space. This allows us to derive a Boolean equation system from the original fixpoint expression and its model.

A modal μ-calculus formula can be represented as an ordered equation system. For example the fixpoint expression $\nu Z_1.[a]\mu Z_2.[b]((Z_1 \wedge Q) \vee Z_2)$ is equivalent to the equation system $\mathcal{E}$: $Z_1 \stackrel{\nu}{=} [a]Z_2$, $Z_2 \stackrel{\mu}{=} [b]((Z_1 \wedge Q) \vee Z_2)$, where the variables are ordered by $Z_1 < Z_2$. Note that in general here the order on the variables is a partial order in contrast to the variable ordering as in definition 1.

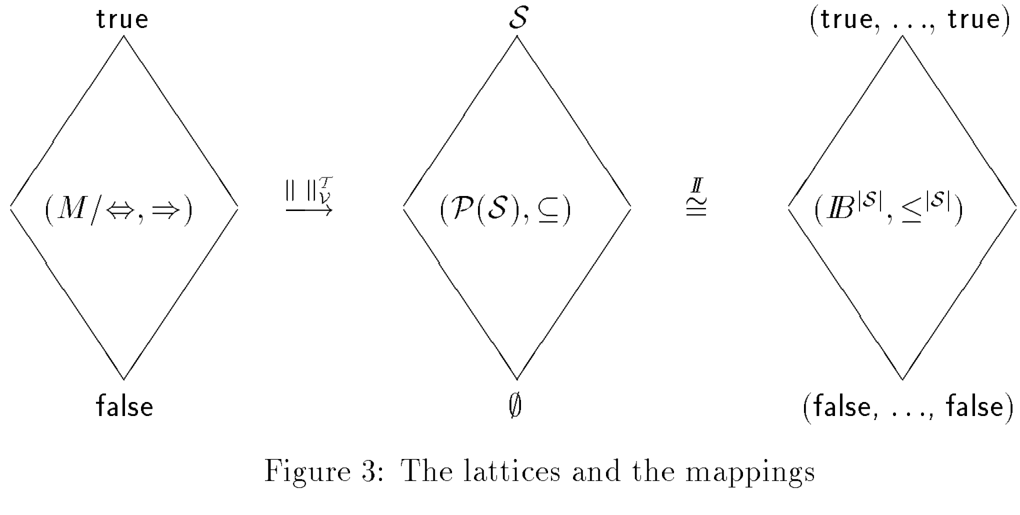


Figure 3: The lattices and the mappings

The transformation is as follows: Recall that M is the set of variable and fixpoint free expressions of the modal μ-calculus, i.e., the expressions of the propositional modal logic. The equivalence classes of M together with the implication ordering form a lattice $(M/\Leftrightarrow, \Rightarrow)$. The powerset of the state space $\mathcal{S} = \{s_1, \ldots, s_n\}$ with the inclusion order forms a complete lattice $(\mathcal{P}(\mathcal{S}), \subseteq)$. The evaluation function $\|\ \|_{\mathcal{V}}^{\mathcal{T}} : M \to \mathcal{P}(\mathcal{S})$ is monotone (and continuous). The extension of the evaluation function from M to fixpoint equations over M maps modal operators $[a], \langle a\rangle$ to set operators $[\![a]\!]^{\mathcal{T}}, \langle\!\langle a\rangle\!\rangle^{\mathcal{T}}$, modal variables to set variables and the logical operators $\wedge, \vee$ to the set operators $\cap, \cup$. Thus we get an equation system over the powerset of the state space. The labels $\{\nu, \mu\}$ and the partial order on the equations remain the same as in the original equation system. Defining $\mathsf{false} \leq \mathsf{true}$ the Boolean lattice $(I\!B^{|\mathcal{S}|}, \leq^{|\mathcal{S}|})$ with pointwise ordering is isomorphic to $(\mathcal{P}(\mathcal{S}), \subseteq)$. The last step leads from a vector valued equation system in $\mathbb{B}^n$ to a Boolean equation system; every vector equation is split into n equations and the operators $[\![a]\!]^{\mathcal{T}}, \langle\!\langle a\rangle\!\rangle^{\mathcal{T}}$ are evaluated.

Altogether a μ-calculus equation $Z \stackrel{\sigma}{=} \Phi$ is mapped inductively to the (not ordered) set of n Boolean equations $\quad \mathbf{I}_Z(s_i) \stackrel{\sigma}{=} \mathbf{I}_\Phi(s_i) \quad$ for $1 \leq i \leq n$, where

$$
\begin{aligned}
\mathbf{I}_Q(s) &= \mathsf{true}, \text{ if } s \in \|Q\|_{\mathcal{V}}^{\mathcal{T}} & \qquad \mathbf{I}_Z(s) &= X_{Z,s} \\
\mathbf{I}_Q(s) &= \mathsf{false}, \text{ if } s \notin \|Q\|_{\mathcal{V}}^{\mathcal{T}} & \qquad \mathbf{I}_{[a]\Phi}(s) &= \bigwedge_{s \xrightarrow{a} s'} \mathbf{I}_\Phi(s') \\
\mathbf{I}_{\Phi_1 \wedge \Phi_2}(s) &= \mathbf{I}_{\Phi_1}(s) \wedge \mathbf{I}_{\Phi_2}(s) & \qquad \mathbf{I}_{\langle a\rangle\Phi}(s) &= \bigvee_{s \xrightarrow{a} s'} \mathbf{I}_\Phi(s') \\
\mathbf{I}_{\Phi_1 \vee \Phi_2}(s) &= \mathbf{I}_{\Phi_1}(s) \vee \mathbf{I}_{\Phi_2}(s) & &
\end{aligned}
$$

Note that the Boolean equations derived in this way do not contain negations or modal operators. A μ-calculus equation system of the size k determines a Boolean equation system of size $k * |\mathcal{S}|$. There is a partial order on the Boolean equations inherited from the partial order on the μ-calculus equations. Two Boolean equations derived from one vector equation are not ordered. Thus the tree-like order on the set equations becomes an acyclic order on the Boolean equations.

We now show that the two algorithms proposed in the previous section can be applied to the Boolean equation systems derived from a modal μ-calculus equation system and a model. There are remaining two open questions: first, whether the partial order on the Boolean equations here matches the linear order on the Boolean equations and variables as in definition 1, and second, whether the solution of a Boolean equation system coincides with the semantics of the modal μ-calculus.

Proposition 4 Given a μ-calculus expression and a transition system let $\underline{A}$ be the corresponding Boolean equation system. On its equations a partial order $<_{\mathcal{E}}$ is defined. For each two extensions of $<_{\mathcal{E}}$ to a linear order $<_l, <_{l'}$ it holds: $\underline{Y}$ is the solution of $\underline{A}$ with the order $<_l$, iff $\underline{Y}$ is the solution of $\underline{A}$ with the order $<_{l'}$.

Proof (Sketch of the Proof)
For unnested fixpoints an order of equations is not relevant for the solution (see [Bek84]). The order of the nested fixpoints is preserved by each extension of the partial order $<_{\mathcal{E}}$ to a linear order. □

Proposition 5 Given a fixpoint expression Φ of the modal μ-calculus and a transition system $\mathcal{T}$ with the initial state s, let $\underline{A}$ be the corresponding Boolean equation system. For the solution $\underline{Y}$ of $\underline{A}$ holds: $Y_{\Phi,s} = \mathsf{true}$, iff Φ holds at s.

Proof It is easy to see that the algorithm of Emerson and Lei [EmL86] calculates the solution as in Definition 2. □

5 Comparison to Other Work and Complexity

The model checking problem encoded as equation system was already treated by several authors [AC88, And92, CDS92, Lar92]. The method presented here differs from these approaches. Roughly speaking, the difference is that they get the solution by approximating sets. This approach was introduced by Emerson and Lei [EmL86] and continued for example by Cleaveland, Dreimüller and Steffen [CDS92]. In [And92] Andersen gives an algorithm based on Boolean equations. However, by representing a Boolean equation system as a graph his basic algorithm applies only for unnested fixpoints. The extension of the global version of his algorithm to the full calculus is due to the fixpoint approximation technique of Emerson and Lei. Also Larsen's algorithm [Lar92] deals with unnested fixpoints.

The Gauß elimination for model checking in its local version is more closely related to the tableau method of Stirling and Walker [StW89] and Cleaveland [Cle90]. In a Boolean equation system a variable is introduced for each pair of a state and a fixpoint formula. Each node in a tableau, a sequent, is a pair of a state and a formula. Hence a Boolean equation can be seen like a reduced form of a subtableau containing only sequents with fixpoint formulae, which is the only relevant part for the structure of the tableau. The top-down construction of the tableau can also be found in the local version of the Gauß elimination. While constructing a tableau the decision which path should be extended is equivalent to the selection of a variable from the top equation and creating the related equation. The condition for a leaf in the tableau of being successful or not corresponds to the Gauß division step: a cycle with a minimal fixpoint is regarded as unsuccessful (false), a cycle with a maximal fixpoint is regarded as successful (true). The advantage of the Gauß elimination over the tableau method has its roots in the bottom-up evaluation. On one hand it spares the introduction of different constants for the same fixpoint expression, on the other hand there is no redundant evaluation of identical subexpressions (subtrees). Altogether the local version of the Gauß elimination for model checking can be regarded as a combination of the top-down strategy of the tableau allowing to explore only the relevant part of the state space, and a bottom-up strategy which avoids recomputation of identical subtrees. The maximal size of a tableau is bounded by $O(b^{(|\Phi| * |\mathcal{S}|)^{f(\Phi)}})$, where b is the maximal branching degree of the transition system, $|\mathcal{S}|$ the number of states, $|\Phi|$ the size of the formula and $f(\Phi)$ the number of fixpoint operators in Φ. For the Gauß elimination the number of derived equations is determined by the size of the state space and the number of fixpoint operators in Φ. Substituting Boolean expressions leads to expressions exponential in the number of equations. The maximal size of the Boolean equation system constructed by the Gauß elimination is bound by $O((b^{a(\Phi)} * |\Phi|)^2 * 2^{|\mathcal{S}| * f(\Phi)})$, where additional to the abbreviations above $a(\Phi)$ is the maximal nesting depth of modal operators in the formula

Φ. Hence it is a natural idea to use the local version of the Gauß elimination for an implementation of the tableau method.

6 Examples

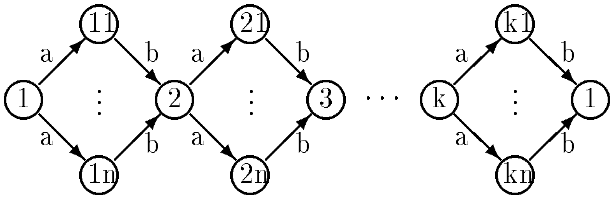
The aim of this section is to demonstrate the possible advantage of the local version of Gauß elimination over a tableau based model checker.

The first both examples are academic ones, without a special meaning. They do not show the advantage of local model checking, because the whole state space has to be traversed. However, they show how our algorithm avoids recomputation of subexpressions, or subtrees resp., whereas the tableau method does not.

In the third example we prove a fairness property for the mutual exclusion algorithm of Peterson.

A prototype of the local version of Gauß elimination was implemented in C++ using BDDs [Bry92] as data structure for Boolean expressions. In the examples here we compared our implementation with a tableau-based model checker as in [StW89] and with the tableau-based model checker incorporated in the Concurrency Workbench (CWB), which uses techniques for avoiding some recomputations. All implementations run on a SUN SPARC2.

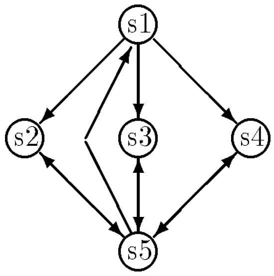
We wish to determine whether $s1 \models_{\mathcal{M}} \nu X.[a]\langle b\rangle X$ (every a-successor has a b-successor and this recursively) holds in:



The example consists of a scalable (n, k)-spindle where the final state is again identified with the start state. It has $kn + k$ states.

The local Gauß elimination creates k equations, each of the form: $X_i \stackrel{\nu}{=} \bigwedge_{j=1..n} X_{i+1 \ mod \ k}$ for $1 \leq i \leq k$ which can be reduced to $X_i \stackrel{\nu}{=} X_{i+1 \ mod \ k}$. It takes k elimination steps to determine the solution. The tableau based model checker as in [StW89] builds a tree with $1 + 2n + 2n^2 + \ldots + 2n^k$ sequents. For this property the number of equations is thus linear in the variable k of the (n, k)-spindle, whereas the size of the tableau is exponential in the variable k.

Does the following $s1 \models_{\mathcal{M}} \nu Z_1.\langle a\rangle \mu Z_2.\langle a\rangle\langle a\rangle Z_2 \vee \langle a\rangle\langle a\rangle Z_1$ hold in (all transitions labelled with a):



This property was proved by our new local model checker with the following results: It created 6 equations and took one second time for the whole procedure. The tableau based model checker was interrupted after having generated more than 22 million (!) tableau sequents.

The model checker of the Concurrency Workbench could cope well with both examples: it "quickly" returned the result. The techniques for avoiding recomputation came in useful. This was not the case in the following example.

We considered the two process mutual exclusion algorithm of Peterson, given in [Wal89]. The property we proved is: "As long as process 1 proceeds, after a request it eventually enters the critical section." In order to detect progress we added "tick" and "tack" dummy actions which alternate each other when process 1 performs some action. Then the property to prove can be formulated as: "if a request comes, then along all paths where ticks and tacks alternate each other, eventually an enter will follow". The μ-calculus formula representing this property is:

$$\begin{aligned}\nu Z_1.([-]Z_1 \wedge \mu Z_2.(\quad & \nu Z_3.([\backslash\mathrm{enter}]((\langle\mathrm{tick}\rangle tt \vee \\ & \nu Z_4.([\backslash\mathrm{enter}]((\langle\mathrm{tack}\rangle tt \vee Z_2) \wedge Z_4)) \wedge Z_3)))))\end{aligned}$$

This rather difficult modal μ-calculus formula is of alternation depth 3 and nesting depth 4. Unfortunately the full discussion of this example exceeds the aim of the paper.

This property, together with our extended Peterson-2 algorithm, was fed to the model checker, with the following result: The model checker came back with a positive result after slightly more than 10 minutes of CPU time. The CWB model checker on the other hand could not compute an answer for the same input within 24 hours elapsed time. During execution our model checker created 156 out of a possible 240 Boolean equations. This example is typical of the results we got from an extensive investigation into several mutual exclusion algorithms with different liveness properties.

7 Conclusion

We presented a novel, algebraic approach for solving Boolean equation systems. As main application model checking in the full modal μ-calculus was intended. In contrast to other approaches using equation systems our method is not based on approximation techniques and backtracking. The method works straightforward by successively eliminating variables and reducing the Boolean equation system, similar to Gauß elimination for linear equation systems. Homogeneous, hierarchical and alternating fixpoints are treated uniformly. Contrary to other techniques Gauß elimination leads to both a global and a local model checking algorithm within one framework. The local version is closely related to the tableau methods, but has a better worst case complexity. An extension to model checking for infinite state spaces is in work.

There exists a prototype implementation of the Gauß elimination using BDDs for the representation of Boolean expressions. Several examples (e.g. fairness properties for mutex algorithms) showed that the local version of our algorithm beats existing tableau methods.

Acknowledgements Frank Wallner implemented the local version of the algorithm. The examples were elaborated together with Dieter Barnard. I thank Javier Esparza, Astrid Kiehn, Bernhard Steffen and Frank Wallner for proof reading and helpful comments.

References

[And92] H. Andersen. "Model Checking and Boolean Graphs." In *Proc. of ESOP'92*, LNCS 582, 1992.

[AC88] A. Arnold, P. Crubille. "A Linear Algorithm to Solve Fixed-Point Equations on Transition Systems." *Information Processing Letters*, vol. 29, 57-66, 1988.

[BCMDH92] J.R. Burch and E.M. Clarke and K.L. McMillan and D.L. Dill and L.J. Hwang. "Symbolic Model Checking: 10^{20} States and Beyond." *Information and Computation*, Vol. 98, pp. 142-170, 1992.

[Bek84] H. Bekic. "Definable operations in general algebras, and the theory of automata and flow charts." In C.B.Jones, editor, *Hans Bekic: Programming Languages and Their Definition*, LNCS 177, 1984.

[Bra92] J. Bradfield. "Verifying Temporal Properties of Systems." Birkhäuser,1992.

[BS91] J. Bradfield, C. Stirling. "Verifying Temporal Properties of Processes." *Proc. of CONCUR'90*, LNCS 458, 1991.

[Bry92] R.E. Bryant. "Symbolic Boolean Manipulation with Ordered Binary-Decision Diagrams." *ACM Computing Surveys*, Vol. 24, No. 3, September 1992.

[CE81] E. M. Clarke and E. A. Emerson. "Design and synthesis of synchronization skeletons using branching time temporal logics." In LNCS 131, pp.52-71, 1981.

[CES86] E. M. Clarke, E. A. Emerson and A. P. Sistla. "Automatic verification of finite-state concurrent systems using temporal logic specifications." In *ACM Trans. on Programming Languages and Systems 8* , pp. 244-263, 1986.

[Cle90] R. Cleaveland. "Tableau-based model checking in the propositional mu-calculus." *Acta Informatica*, Vol. 27, pp. 725-747, 1990.

[ClS91] R. Cleaveland and B. Steffen. "A Linear-Time Model Checking Algorithm for the Alternation-Free Modal Mu-Calculus." In *Proc. of CAV'91*, LNCS 575, 1992.

[CDS92] R. Cleaveland, M. Dreimüller and B. Steffen. "Faster Model Checking for the Modal Mu-Calculus." In *Proc. of CAV'92*, LNCS 663, 1993.

[EmL86] E.A. Emerson and C.-L. Lei. "Efficient model checking in fragments of the propositional mu-calculus." In *Proc. of LICS'86*, Computer Society Press, 1986.

[Koz83] D. Kozen. "Results on the Propositional μ-Calculus." *TCS*, Vol. 27, 1983, pp. 333-354.

[Lar92] K. Larsen. "Efficient Local Correctness Checking." In *Proc. of CAV'92*, LNCS 663, 1993.

[LBCJM94] D. E. Long, A. Browne, E. M. Clarke, S. Jha, W. R. Marrero. "An improved algorithm for the evaluation of fixpoint expressions." In *Proc. of CAV'94*, LNCS 818, 1994.

[Mad92] A. Mader. "Tableau Recycling." In *Proc. of CAV'92*, LNCS 663, 1993.

[Rud74] S. Rudeanu. "Boolean Functions and Equations." North-Holland Publishing Company, 1974.

[Sti92] C. Stirling. "Modal and Temporal Logics." In *Handbook of Logic in Computer Science*, Oxford University Press, 1992.

[StW89] C. Stirling and D. Walker. "Local model checking in the modal mu-calculus." In *Proc. of TAPSOFT'89*, LNCS 351, 1989.

[Tar55] A. Tarski. " A Lattice Theoretical Fixpoint Theorem and its Applications." In *Pacific Journal of Mathematics*, 5, 1955.

[VeL92] B. Vergauwen and J. Lewi "A linear algorithm for solving fixed-point equations on transition systems" In *Proc. of CAAP'92*, LNCS 581, 1992.

[Wal89] D. Walker. "Automated Analysis of Mutual Exclusion Algorithms using CCS." Technical Report ECS-LFCS-89-91, University of Edinburgh, 1991.

[Xin92] Liu Xinxin. "Specification and Decomposition in Concurrency." PhD Thesis, Aalborg University Center, Denmark, 1992.

Appendix: Proofs

Proposition 1 For the solution Y of the equation system $X \stackrel{\sigma}{=} A(X)$ consisting of one single equation it holds:

$$Y = \begin{cases} A(\mathsf{false}) & \text{if } \sigma = \mu \\ A(\mathsf{true}) & \text{if } \sigma = \nu \end{cases}$$

Proof For $\sigma = \mu$ there are three cases:

1) $A(X) = \mathsf{true}$, i.e. the evaluation of the expression A is independent of the valuation of the free variable X. Then the solution is $Y = \mathsf{true}$.
2) $A(X) = \mathsf{false}$, i.e. the evaluation of the expression A is independent of the valuation of the free variable X. Then the solution is $Y = \mathsf{false}$.
3) In the remaining case because of monotonicity of $A(X)$ the following holds: $A(\mathsf{false}) = \mathsf{false}$ and $Y = \mathsf{false}$ is the solution.

Analogously it holds $Y = A(\mathsf{true})$ that is the solution of $X \stackrel{\nu}{=} A(X)$. □

Often the solution of $X \stackrel{\sigma}{=} A(X)$ will be denoted by the expression $\sigma X.A(X)$.

For the proof of propositions 2 and 3 we need the following property of the solution.

Proposition 6 The solution $\underline{Y}$ of the Boolean equation system $\underline{E}$ is an extremal point of $\underline{E}$, i.e., for $1 \leq i \leq n$ holds:
Y_i is solution of $X_i \stackrel{\sigma_i}{=} A_i(Y_1, \ldots, Y_{i-1}, X_i, Y_{i+1}, \ldots, Y_n)$.

Proof trivial $\square$

Proposition 2 $\underline{Y}$ is the solution of the Boolean equation system $\underline{E}$ of the form $\underline{X} \stackrel{\sigma}{=} \underline{A}(\underline{X})$, iff $\underline{Y}$ also is the solution of the modified Boolean equation system $\underline{F}$, where the equations are of the following form:

$$\begin{array}{lcl} X_1 & \stackrel{\sigma_1}{=} & A_1(\mathsf{b}_1, X_2, \quad \ldots \quad, X_n) \\ \vdots & & \\ X_i & \stackrel{\sigma_i}{=} & A_i(X_1, \ldots, X_{i-1}, \mathsf{b}_i, X_{i+1}, \ldots, X_n) \\ \vdots & & \\ X_n & \stackrel{\sigma_n}{=} & A_n(X_1, \quad \ldots \quad, X_{n-1}, \mathsf{b}_n) \end{array} \qquad \text{where} \quad \mathsf{b}_i = \begin{cases} \mathsf{true} & \text{if } \sigma_i = \nu \\ \mathsf{false} & \text{if } \sigma_i = \mu \end{cases}$$

Proof First we sketch the proof scheme, which will be the same for this and the next proof. For reasons of readability we introduce another abbreviation: $\underline{E}^{(i+1)}$ denotes $\underline{E}^{(i+1)}[Y_1/X_1, \ldots, Y_i/X_i]$, and analogously $\underline{E}^{(i)}$ denotes $\underline{E}^{(i)}[Y_1/X_1, \ldots, Y_{i-1}/X_{i-1}]$. The same holds for $\underline{F}^{(i+1)}$ and $\underline{F}^{(i)}$.
The proof is based on induction on i. The hypothesis for an induction step is: $\underline{Y}^{(i+1)}$ is solution of $\underline{E}^{(i+1)}$, iff $\underline{Y}^{(i)}$ is solution of $\underline{F}^{(i+1)}$. Then the conclusion of an induction step will be: $\underline{Y}^{(i)}$ is solution of $\underline{E}^{(i)}$, iff $\underline{Y}^{(i)}$ is solution of $\underline{F}^{(i)}$. For this purpose we have to show the following: The solution of $\underline{E}^{(i)}$ is a fixpoint of $\underline{F}^{(i)}$, i.e. the solution of $\underline{F}^{(i)}$ is lower or equal to the solution of $\underline{E}^{(i)}$.
On the other hand we have to show that the solution of $\underline{F}^{(i)}$ is also a fixpoint of $\underline{E}^{(i)}$, i.e. the solution of $\underline{E}^{(i)}$ is at most smaller than the solution of $\underline{F}^{(i)}$. From these we can conclude that both solutions must be the same.
(i) induction basis: Y_n is the solution of $\underline{E}^{(n)}$, iff Y_n is the solution of $\underline{F}^{(n)}$. Suppose Y_n is the solution of $X_n \stackrel{\sigma_n}{=} A_n(Y_1, \ldots, Y_{n-1}, X_n)$. According to proposition 1 for the solution Y_n holds: $Y_n = A_n(Y_1, \ldots, Y_{n-1}, \mathsf{b}_n)$ and hence Y_n is also the solution of $\underline{F}^{(n)}$. With the same argument the converse holds.
(ii) induction step: for all i with $1 \leq i < n$ we now prove that the solution of the one equation system $\underline{E}^{(i)}$ is a fixpoint of $\underline{F}^{(i)}$ and conversely. Here we show a stronger proposition, namely that every extremal point of one equation system is also a fixpoint of the other one. Because the solution is an extremal point, we can conclude the desired proposition.
Suppose $\underline{Y}^{(i)}$ is an extremal point of $\underline{E}^{(i)}$. Then for $i \leq k \leq n$ holds:

$$\begin{array}{lcll} Y_k & = & \sigma_k X_k . A_k(Y_1, \ldots, Y_{k-1}, X_k, Y_{k+1}, \ldots, Y_n) & \underline{Y}^{(i)} \text{ is extremal point of } \underline{E}^{(i)} \\ & = & A_k(Y_1, \ldots, Y_{k-1}, \mathsf{b}_k, Y_{k+1}, \ldots, Y_n) & \text{proposition 1} \end{array}$$

Hence $\underline{Y}^{(i)}$ is also a fixpoint of $\underline{F}^{(i)}$.
Conversely suppose $\underline{Y}^{(i)}$ is an extremal point of $\underline{F}^{(i)}$. Then for $i \leq k \leq n$ it holds:

$$\begin{array}{lcll} Y_k & = & \sigma_k X_k . A_k(Y_1, \ldots, Y_{k-1}, \mathsf{b}_k, Y_{k+1}, \ldots, Y_n) & \underline{Y}^{(i)} \text{ is extrem. point of } \underline{F}^{(i)} \\ & = & A_k(Y_1, \ldots, Y_{k-1}, \mathsf{b}_k, Y_{k+1}, \ldots, Y_n) & X_k \text{ is not free in this expr.} \\ & = & \sigma_k X_k . A_k(Y_1, \ldots, Y_{k-1}, X_k, Y_{k+1}, \ldots, Y_n) & \text{proposition 1} \\ & = & A_k(Y_1, \ldots, Y_{k-1}, Y_k, Y_{k+1}, \ldots, Y_n) & \text{ev. extrem. point is a fixp.} \end{array}$$

Hence $\underline{Y}^{(i)}$ is also a fixpoint of $\underline{E}^{(i)}$.
With these arguments we now can apply the induction step n times, and altogether we get the proposition. $\square$

Proposition 3 The Boolean vector $\underline{Y}$ is the solution of the equation system $\underline{E}$, iff it is the solution of the equation system $\underline{G}$, where $\underline{G}$ is the modified equation system:

$$\begin{array}{rcl} X_1 & \stackrel{\sigma_1}{=} & A_1 \quad (X_1, \ldots, X_{n-1}, A_n(X_1, \ldots, X_{n-1}, \mathsf{b}_n)) \\ \vdots & & \\ X_{n-1} & \stackrel{\sigma_{n-1}}{=} & A_{n-1}(X_1, \ldots, X_{n-1}, A_n(X_1, \ldots, X_{n-1}, \mathsf{b}_n)) \\ X_n & \stackrel{\sigma_n}{=} & A_n \quad (X_1, \ldots, X_{n-1}, \mathsf{b}_n) \end{array}$$

Proof The proof is based on the same scheme as the one of proposition 2.
We will use the analogous abbreviations as in proposition 2.
(i) induction basis:
Y_n is the solution of $\underline{E}^{(n)}$, iff Y_n is the solution of $\underline{G}^{(n)}$.
Suppose Y_n is the solution of $X_n \stackrel{\sigma_n}{=} A_n(Y_1, \ldots, Y_{n-1}, X_n)$. According to proposition 1 the following holds: $Y_n = A_n(Y_1, \ldots, Y_{n-1}, \mathsf{b}_n)$ and hence Y_n is also the solution of $\underline{G}^{(n)}$. The converse holds with the same argument.
(ii) induction step:
For all i with $1 \leq i < n$ each extremal point of $\underline{G}^{(i)}$ is a fixpoint of $\underline{E}^{(i)}$ and conversely.
Suppose $\underline{Y}^{(i)}$ is an extremal point of $\underline{G}^{(i)}$.

$$\begin{array}{rcll} Y_n & = & A_n(Y_1, \ldots, Y_{n-1}, \ \mathsf{b}_n) & \underline{Y} \text{ is a fixpoint of } \underline{G} \\ & = & \sigma_n X_n . A_n(Y_1, \ldots, Y_{n-1}, X_n) & \text{proposition 1} \\ (*) & = & A_n(Y_1, \ldots, Y_{n-1}, \ Y_n) & \text{every extremal point is also a fixpoint} \end{array}$$

and for $i \leq k \leq n$:

$$\begin{array}{rcll} Y_k & = & A_k(Y_1, \ldots, Y_{n-1}, A_n(Y_1, \ldots, Y_{n-1}, \ \mathsf{b}_n)) & \underline{Y}^{(i)} \text{ is a fixpoint of } \underline{G}^{(i)} \\ & = & A_k(Y_1, \ldots, Y_{n-1}, \ Y_n) & \text{with (*)} \end{array}$$

Hence $\underline{Y}^{(i)}$ is also a fixpoint $\underline{E}^{(i)}$.
Now suppose $\underline{Y}^{(i)}$ is an extremal point of $\underline{E}^{(i)}$.

$$\begin{array}{rcll} Y_n & = & \sigma_n X_n . A_n(Y_1, \ldots, Y_{n-1}, X_n) & \underline{Y}^{(i)} \text{ is an extremal point of } \underline{E}^{(i)} \\ (**) & = & A_n(Y_1, \ldots, Y_{n-1}, \ \mathsf{b}_n) & \text{proposition 1} \end{array}$$

and for $i \leq k \leq n$:

$$\begin{array}{rcll} Y_k & = & A_k(Y_1, \ldots, Y_{n-1}, Y_n) & \underline{Y} \text{ is a fixpoint of } \underline{E} \\ & = & A_k(Y_1, \ldots, Y_{n-1}, A_n(Y_1, \ldots, Y_{n-1}, \mathsf{b}_n)) & \text{with (**)} \end{array}$$

Hence $\underline{Y}^{(i)}$ is also a fixpoint of $\underline{G}^{(i)}$.
Now we can apply the induction step n times, and altogether we get the proposition. □