ELSEVIER

Information Processing Letters 59 (1996) 303-308

Information Processing Letters

Fast and simple nested fixpoints

Helmut Seidl [1]

FB IV - Informatik, Universität Trier, D-54286 Trier, Germany

Received 28 February 1996; revised 22 July 1996
Communicated by H. Ganzinger

Abstract

We give an alternative proof of the result of Long et al. (1994) that nested fixpoint expressions e of alternation depth $d > 1$ can be evaluated over a complete lattice of height h in time $O(d \cdot (h \cdot |e|/(d-1))^{\lceil d/2 \rceil + 1})$. The advantage of our proof is that it is both extremely short and extremely simple.

Keywords: Evaluation of fixpoint expressions; Hierarchical systems of equations; Model-checking

1. Introduction

Since Kozen's seminal paper [11] on modal μ-calculus in 1983, this logic has been widely used for the specification of properties of concurrent processes (see, e.g., [13,14]). Especially, the model-checking problem, i.e., the question to determine whether or not a given system satisfies a formula, has attracted attention.

In [8], Emerson and Lei presented an algorithm where the exponent in the worst-case complexity is given by $d + 1$ where d is the *alternation depth* of the formula to be verified. Their method has been refined by Cleaveland et al. [7]. For formulas of alternation depth 1, even linear algorithms have been proposed (see, e.g., [1,2,6]). For other important fragments of μ-calculus, Emerson et al. exhibited polynomial algorithms [9] which were slightly improved by Bhat and Cleaveland [4].

A major improvement in the general case has been achieved by Long et al. [12]. They pointed out that the exponent in the worst case complexity can be lowered even to "about" $d/2 + 1$. Their paper explains the method for certain special kinds of formulas. Nevertheless, the argumentation is non-trivial and involved. Their result itself, however, is (at least in our opinion) so important that a simpler explanation is worthwhile. In fact, our argumentation turns out to be almost trivial. Our idea consists in removing one sort of fixpoint iterations simply by *syntactical iteration*, i.e., unfolding. The whole art consists in doing so without overly increasing the size of the resulting system.

To make our idea precise, this short note is organized as follows. Section 2 introduces basic notions concerning (hierarchical) systems of equations. Section 3 deals with the case of alternation depth 1. Section 4 formalizes our idea of syntactical removal of fixpoint iterations and derives the main result. Section 5 contains a brief discussion of our method and concludes.

[1] Email: seidl@psi.uni-trier.de.

2. A fixpoint calculus

In essence, the model-checking problem for μ-formulas boils down to the evaluation of a fixpoint expression like

$$\mu x_1.f_1\,(\mu x_2.f_2\,(h\,x_1, x_2), \nu y.g\,(h\,x_1, y))$$

over some finite complete lattice. Instead of defining such kinds of expressions (together with their semantics) formally, we prefer to introduce the slightly more flexible concept of *hierarchical systems of equations*. In a subsequent example, we hope to convince the reader that fixpoint expressions very easily can be translated into our formalism.

Assume we are given a finite complete lattice D of height h, i.e., the maximal number of elements in a strictly increasing sequence

$$d_0 \sqsubset d_1 \sqsubset \cdots \sqsubset d_r$$

in D equals $h+1$. Let Ω denote a set of operator symbols where each $f \in \Omega$ denotes a monotonic function $[\![f]\!] : D^k \to D$ for some $k \geqslant 1$.

As right-hand sides of equations we allow expressions built up from formal variables from some set $\mathcal{Y}$ and constants by application of operators from Ω. The set of all these expressions is denoted by $\mathcal{E}_{\Omega\cup D}(\mathcal{Y})$. Every expression $e \in \mathcal{E}_{\Omega\cup D}(\mathcal{Y})$ denotes a function $[\![e]\!] : (\mathcal{Y} \to D) \to D$. This function is monotonic since all operations in Ω are monotonic.

A *hierarchical* system of equations with free variables from $\mathcal{F}$ is a pair $(S, \mathcal{H})$. S is the finite basic set of equations $x = e_x$, $x \in \mathcal{X}$, where for every x, e_x is an expression in $\mathcal{E}_{\Omega\cup D}(\mathcal{X} \cup \mathcal{F})$, and $\mathcal{H}$ is a *hierarchy* on $\mathcal{X}$. A hierarchy consists of a sequence $\mathcal{H} = (\langle\mathcal{X}_k, \lambda_k\rangle, \ldots, \langle\mathcal{X}_1, \lambda_1\rangle)$ of mutually disjoint nonempty sets $\mathcal{X}_j$ where $\mathcal{X} \supseteq \mathcal{X}_1 \cup \cdots \cup \mathcal{X}_k$ together with qualifications $\lambda_j \in \{\mu, \nu\}$. Intuitively, hierarchy $\mathcal{H}$ on S is introduced in order to reflect the nesting of scopes of variables within which fixpoint iteration is performed. The set $\mathcal{X}_0 = \mathcal{X} \setminus (\mathcal{X}_1 \cup \cdots \cup \mathcal{X}_k)$ represents the "non-recursive" *auxiliary* variables. We assume a suitable linear ordering "$\leqslant$" on $\mathcal{X}_0$ such that $y \in \mathcal{X}_0$ occurs in the right-hand side e_x, $x \in \mathcal{X}_0$, only if $y < x$. Fixpoint iteration is (potentially) performed on the variables in $\mathcal{X}_1 \cup \cdots \cup \mathcal{X}_k$. These are therefore also called *iteration variables*.

Example 1. Assume we are given a fixpoint expression like

$$\phi \equiv \mu x_1.f_1\,(\mu x_2.f_2\,(h\,x_1, x_2), \nu y.g\,(h\,x_1, y))$$

A representation of this formula by a hierarchical system is obtained by introducing an extra equation for each fixpoint subexpression. For our example this gives set S consisting of:

$$x_1 = f_1\,(x_2, y)$$
$$x_2 = f_2\,(h\,x_1, x_2)$$
$$y = g\,(h\,x_1, y)$$

Hierarchy $\mathcal{H}$ now is obtained by dividing the set of fixpoint variables of ϕ into "layers" within which only fixpoints of the same kind occur. Thus for our example formula ϕ, $\mathcal{H} = (\langle\mathcal{X}_2, \mu\rangle, \langle\mathcal{X}_1, \nu\rangle)$ where $\mathcal{X}_1 = \{y\}$ and $\mathcal{X}_2 = \{x_1, x_2\}$.

Auxiliary variables can be introduced for returning the result for the whole formula (in case it does not have a fixpoint on top level) or to remove multiple occurrences of the same subexpression. In our example, we could introduce a fresh auxiliary variable z to receive the value for $h\,x_1$. This results in the following set of equations:

$$x_1 = f_1\,(x_2, y)$$
$$x_2 = f_2\,(z, x_2)$$
$$y = g\,(z, y)$$
$$z = h\,x_1$$

Usually, if $\mathcal{H}$ is understood, we write S for the hierarchical system. We also denote the fact that $\mathcal{X}_j$ is qualified with λ_j by $\mathcal{X}_j : \lambda_j$.

Fix some $0 \leqslant j \leqslant k$, and let $\mathcal{X}' = \mathcal{X}_0 \cup \cdots \cup \mathcal{X}_j$. Then the jth *subsystem* S_j of S is given by the base set of equations $x = e_x$, $x \in \mathcal{X}'$, together with hierarchy $\mathcal{H}_j = (\langle\mathcal{X}_j, \lambda_j\rangle, \ldots, \langle\mathcal{X}_1, \lambda_1\rangle)$. Note that the free variables of S_j are contained in $\mathcal{F} \cup \mathcal{X}_{j+1} \cup \cdots \cup \mathcal{X}_k$.

An *environment* ρ for S is a mapping $\rho : \mathcal{F} \to D$. The *solution* of S relative to environment ρ in D, denoted by $[\![S]\!]\,\rho$, is a mapping $\mathcal{X} \to D$. It is defined by induction on length k of hierarchy $\mathcal{H}$. If $k = 0$, then no fixpoint iteration is necessary. Values $[\![S]\!]\,\rho\,x, x \in \mathcal{X}_0$, are obtained by successively evaluating right-hand sides according to ordering "$\leqslant$". Formally, let $\mathcal{X}_0 =$

$\{x_1, \ldots, x_m\}$ where $x_1 < \cdots < x_m$, and assume $\beta : \{x_1, \ldots, x_{j-1}\} \to D$, given by $\beta x_i = [\![S]\!]\ \rho\ x_i$, $i < j$, has already been defined. Then $[\![S]\!]\ \rho\ x_j = [\![e_{x_j}]\!]\ (\rho + \beta)$. Note that we use the "+"-operator to combine two functions with disjoint domains into one.

For $k > 0$, let $\mathcal{X}' = \mathcal{X} \backslash \mathcal{X}_k$ and S_{k-1} denote the $(k-1)$th subsystem of S. Consider monotonic function $G : (\mathcal{X}_k \to D) \to \mathcal{X}_k \to D$ given by $G\ \beta\ x = [\![e_x]\!]\ (\rho + \beta + [\![S_{k-1}]\!]\ (\rho + \beta))$. In case $\mathcal{X}_k$ is qualified as μ, let $\bar{\beta}$ denote the least fixpoint of G. Otherwise, let $\bar{\beta}$ denote the greatest fixpoint of G. Then $[\![S]\!]\ \rho$ is defined by $[\![S]\!]\ \rho\ x = \bar{\beta}\ x$ if $x \in \mathcal{X}_k$ and $[\![S]\!]\ \rho\ x = [\![S_{k-1}]\!]\ (\rho + \bar{\beta})\ x$ if $x \in \mathcal{X}'$.

Fact 2. *Assume $k > 1$ and S is a system of equations with hierarchy $\mathcal{H}$ given by*

$$(\langle \mathcal{X}_k, \lambda_k \rangle, \langle \mathcal{X}_{k-1}, \lambda_{k-1} \rangle, \ldots, \langle \mathcal{X}_1, \lambda_1 \rangle).$$

If $\lambda_k = \lambda_{k-1}$ then for every environment ρ, S together with $\mathcal{H}$ has the same solution as S together with hierarchy

$$(\langle \mathcal{X}_k \cup \mathcal{X}_{k-1}, \lambda_k \rangle, \langle \mathcal{X}_{k-2}, \lambda_{k-2} \rangle, \ldots, \langle \mathcal{X}_1, \lambda_1 \rangle).$$

It follows that we can always remove successive occurrences of μ's or ν's in the hierarchy until we end up with a hierarchy $(\langle \mathcal{X}_k, \lambda_k \rangle, \ldots, \langle \mathcal{X}_1, \lambda_1 \rangle)$ where $\lambda_j \neq \lambda_{j-1}$ for all $j > 1$. Systems with this property are called *normalized*. The length k of the hierarchy of a normalized system is also called its *alternation depth*. Note that we deviate here from the definitions in [4], [7] or [8] for formulas. the definitions there reflect the possibility of extracting closed formulas and computing their meanings once and for all beforehand. It is for simplicity that we refrain from integrating a similar idea for hierarchical systems of equations into our definition and algorithm but remark that such an optimization is clearly possible as well.

3. Systems with alternation depth 1

In order to formally estimate computational costs, let us assume that our algorithm is implemented on a Random Access Machine with uniform cost measure, i.e., storing and retrieving elements from D, comparing elements from D for equality and applying operators from Ω are all counted for 1. Accordingly, we define the size of system S as $|S| = \sum_{x \in \mathcal{X}} 1 + |e_x|$. In case the alternation depth of system S equals 1, computation of the solution reduces to the task of computing the least (respectively greatest) solution of an ordinary system of equations over complete lattice D. This can be done by a standard worklist algorithm. We conclude:

Proposition 3. *Given a system S of alternation depth 1 and environment ρ for the free variables in S, the solution of S can be computed in time $O(h \cdot |S|)$.* □

4. Syntactical removal of fixpoint iterations

The idea of our algorithm is based on the trivial observation that in a lattice of height h, the least fixpoint is reached after at most h iterations. Define the nth iterate of g on d by $g^0\ d = d$ and for $n > 0$, $g^n\ d = g\ (g^{n-1}\ d)$. We have:

Observation 4. *If $g : D \to D$ is monotonic then the least fixpoint (respectively greatest fixpoint) of g is given by $g^h\ \bot$ (respectively $g^h\ \top$).*

The same observation holds true for monotonic mapping $g : (\mathcal{Y} \to D) \to (\mathcal{Y} \to D)$ if exponent h is replaced with exponent $q = h \cdot \#\mathcal{Y}$ where $\#\mathcal{Y}$ denotes the cardinality of set $\mathcal{Y}$. So let S denote a normalized system of alternation depth $k+1$ with hierarchy $(\langle \mathcal{X}_{k+1}, \lambda_{k+1} \rangle, \ldots, \langle \mathcal{X}_1, \lambda_1 \rangle)$. For every $q \geqslant 1$ and $\beta : \mathcal{X}_{k+1} \to D$, we construct system $S\langle \beta, q \rangle$ with alternation depth k. $S\langle \beta, q \rangle$ is obtained from S by (syntactically) unrolling the iteration on the variables in $\mathcal{X}_{k+1}$.

Define $\mathcal{X}'' = \{\langle x, j \rangle \mid x \in \mathcal{X},\ 0 \leqslant j < q\}$, and $\mathcal{X}' = \mathcal{X} \cup \mathcal{X}''$. $S\langle \beta, q \rangle$ consists of the equations $x = e'_x$, $x \in \mathcal{X}'$, together with hierarchy $(\langle \mathcal{X}'_k, \lambda_k \rangle, \ldots, \langle \mathcal{X}'_1, \lambda_1 \rangle)$ where for $j = 1, \ldots, k$, $\mathcal{X}'_j = \mathcal{X}_j \cup \{\langle x, i \rangle \mid x \in \mathcal{X}_j,\ 0 \leqslant i < q\}$.

The right-hand sides e'_x of the equations are defined as follows.

If $x \in \mathcal{X}_{k+1}$, then

- e'_x is obtained from e_x by replacing each occurrence of variable $y \in \mathcal{X}$ with $\langle y, q-1 \rangle$;
- $e'_{\langle x,0 \rangle} = \beta\ x$, and
- for $j > 0$, $e'_{\langle x,j \rangle}$ is obtained from e_x by replacing each occurrence of variables y with $\langle y, j-1 \rangle$.

If $x \in \mathcal{X} \setminus \mathcal{X}_{k+1}$, then
- $e'_x \equiv e_x$, and
- for all j, $e'_{\langle x,j\rangle}$ is obtained from e_x by replacing each occurrence of $y \in \mathcal{X}$ with $\langle y, j\rangle$.

Intuitively, $S\langle\beta, q\rangle$ is obtained by taking q extra copies of S. The variables $\langle x, j\rangle$, $x \in \mathcal{X}_{k+1}$, are meant to receive the jth iteration for variable x. $\langle x, 0\rangle$ receives the start value βx for the iteration whereas for $j > 0$, the value of $\langle x, j\rangle$ is computed relative to the values of $\langle y, j-1\rangle$, i.e., the former approximation. Finally, variables x receive the values of the qth approximation.

Example 5. Consider the system of equations from our last example and assume we are going to compute its solution over lattice $D = \{0 \sqsubset 1\}$ with height 1.

Removing least fixpoint iteration on $\{x_1, x_2\}$ results in:

$$\begin{aligned}
x_1 &= f_1\,(\langle x_2, 1\rangle, \langle y, 1\rangle)\\
x_2 &= f_2\,(\langle z, 1\rangle, \langle x_2, 1\rangle)\\
\langle x_1, 1\rangle &= f_1\,(\langle x, 0\rangle, \langle y, 0\rangle)\\
\langle x_2, 1\rangle &= f_2\,(\langle z, 0\rangle, \langle x, 0\rangle)\\
\langle x_1, 0\rangle &= \perp\\
\langle x_2, 0\rangle &= \perp\\
y &= g\,(z, y)\\
\langle y, 1\rangle &= g\,(\langle z, 1\rangle, \langle y, 1\rangle)\\
\langle y, 0\rangle &= g\,(\langle z, 0\rangle, \langle y, 0\rangle)\\
z &= h\,x_1\\
\langle z, 1\rangle &= h\,\langle x_1, 1\rangle\\
\langle z, 0\rangle &= h\,\langle x, 0\rangle
\end{aligned}$$

with hierarchy $\mathcal{H} = (\langle \mathcal{X}_1, \nu\rangle)$ where $\mathcal{X}_1 = \{y, \langle y, 1\rangle, \langle y, 0\rangle\}$. Note that variables x_i, $\langle x_i, j\rangle$ now have become auxiliary.

We have:

Proposition 6. *Let ρ be an environment for S (and hence also for each $S\langle\beta, q\rangle$). Then the following holds:*
(1) $|S\langle\beta, q\rangle| \leqslant (q+1) \cdot |S|$.
(2) *$S\langle\beta, q\rangle$ can be constructed from S in time* $O((q+1) \cdot |S|)$.
(3) *If $q = h \cdot \#\mathcal{X}_{k+1}$ and $\mathcal{X}_{k+1} : \mu$ then $[\![S]\!]\ \rho\ x = [\![S\langle\underline{\perp}, q\rangle]\!]\ \rho\ x$ for all $x \in \mathcal{X}$ where variable assignment $\underline{\perp}$ maps every $x \in \mathcal{X}_{k+1}$ to $\perp$.*
In case $\mathcal{X}_{k+1} : \nu$, the same holds true if $\underline{\perp}$ is replaced with $\underline{\top}$, mapping every $x \in \mathcal{X}_{k+1}$ to $\top$.

Proof. We only consider assertion (3). Let $\mathcal{X}' = \mathcal{X} \setminus \mathcal{X}_{k+1}$ and S_k denote the kth subsystem of S. Consider monotone function $G : (\mathcal{X}_{k+1} \to D) \to \mathcal{X}_{k+1} \to D$ given by $G\ \beta\ x = [\![e_x]\!]\,(\rho{+}\beta{+}[\![S_k]\!]\,(\rho{+}\beta))$. Without loss of generality $\mathcal{X}_{k+1}$ is qualified as μ. Then $[\![S]\!]\,\rho$, restricted to $\mathcal{X}_{k+1}$, equals the least fixpoint $\overline{\beta}$ of G. Since D has height h, $\overline{\beta} = G^q \underline{\perp}$ for $q = h \cdot \#\mathcal{X}_{k+1}$.

Now consider hierarchical system $S' = S\langle\underline{\perp}, q\rangle$. Assertion (3) follows if we were able to prove for every $0 \leqslant j \leqslant q$ (for simplicity, we introduce $\langle x, q\rangle$ as a renaming of x):
(1) If $x \in \mathcal{X}_{k+1}$ then $[\![S']\!]\ \rho\ \langle x, j\rangle = G^j \underline{\perp}\, x$.
(2) If $x \notin \mathcal{X}_{k+1}$ then $[\![S']\!]\ \rho\ \langle x, j\rangle' = [\![S_k]\!]\,(\rho + G^j \underline{\perp})\, x$.

It is not difficult to see that claim (2) holds for j whenever claim (1) holds for i. Therefore, we concentrate on a proof of claim (1).

We proceed by induction on j. For $j = 0$, claim (1) holds by definition. Now assume $j > 0$. Then we know that $[\![S']\!]\ \rho\ \langle x, j\rangle = [\![e_{\langle x,j\rangle}]\!]\,(\rho + [\![S']\!]\,\rho) = [\![e_x]\!]\,(\rho + \beta)$, where $\beta y = [\![S']\!]\ \rho\ \langle y, j-1\rangle$. By the induction hypothesis, $\beta y = G^{j-1}\ \underline{\perp}\ y$ if $y \in \mathcal{X}_{k+1}$ and $\beta y = [\![S_k]\!]\ (\rho + G^{j-1}\underline{\perp})\ y$ otherwise. But then, by the definition of G, $[\![S']\!]\ \rho\ \langle x, j\rangle = [\![e_x]\!]\,(\rho + G^{j-1}\underline{\perp} + [\![S_k]\!]\,(\rho + G^{j-1}\underline{\perp})) = G^j\ \underline{\perp}\ x$ for every $x \in \mathcal{X}_{k+1}$ which we wanted to prove. □

Note that syntactical unrolling of fixpoint computations is as well possible for sets $\mathcal{X}_j$, $j < k+1$. To eliminate iteration on $\mathcal{X}_j$, we simply replace subsystem S_j with system $S_j\langle\beta, q\rangle$ where $q = h \cdot \#\mathcal{X}_j$ and $\beta = \underline{\perp}$ if $\mathcal{X}_j : \mu$ and $\beta = \underline{\top}$ otherwise.

We combine Fact 2 together with Proposition 6 to derive our main theorem:

Theorem 7. *Assume S is a normalized hierarchical system of alternation depth $d > 1$ with n iteration variables. Then there exists a system S' of alternation depth 1 with identical set of free variables such that the following holds:*
(1) *if d is even,* $|S'| \leqslant (h \cdot n/d + 1)^{d/2} \cdot |S|$;
if d is odd, $|S'| \leqslant (h \cdot n/(d-1) + 1)^{(d+1)/2} \cdot |S|$;

(2) *$|S'|$ can be constructed in time proportional to its size;*
(3) *$[\![S]\!]\ \rho\ x = [\![S']\!]\ \rho\ x$ for every environment ρ and every variable x of S.*

Proof. Repeatedly applying Proposition 6 to the subsystems of S we either may remove from the hierarchy all subsets qualified with ν or all subsets qualified with μ. Subsequent normalization according to Fact 2 yields the desired system S'. It remains to estimate the size of S'.

For the following calculations we first recall that for all $x_1, \ldots, x_r \in \mathbb{N}$, $x_1 + \cdots + x_r \leqslant m$ implies that $x_1 \cdot \ldots \cdot x_r \leqslant (m/r)^r$.

Let $n_1, \ldots, n_s$ be the sequence of sizes of the sets in the hierarchy qualified as ν and $m_1, \ldots, m_t$ the corresponding sequence for sets qualified as μ. If we remove all sets qualified ν from the hierarchy, we obtain from Proposition 6:

$$|S'| \leqslant (n_1 h + 1) \cdots (n_s h + 1) \cdot |S|.$$

Likewise, if we remove all sets qualified μ from the hierarchy, we obtain

$$|S'| \leqslant (m_1 h + 1) \cdots (m_t h + 1) \cdot |S|.$$

We may remove that kind of qualification where sizes sum up to $\leqslant n/2$. Without loss of generality assume that this is the case for ν, i.e., $n_1 + \cdots + n_s \leqslant n/2$.

First assume d is even. Then $s = t = d/2$. Hence,

$$(n_1 h + 1) \cdots (n_s h + 1) \leqslant \left(\frac{nh}{2s} + 1\right)^s \leqslant \left(\frac{nh}{d} + 1\right)^{d/2}.$$

Now consider the case where d is odd. Since $(d-1)/2 \leqslant s, t \leqslant (d+1)/2$, we have:

$$(n_1 h + 1) \cdots (n_s h + 1) \leqslant \left(\frac{nh}{2s} + 1\right)^s \leqslant \left(\frac{nh}{d-1} + 1\right)^{(d+1)/2}$$

and our assertion follows. □

Finally applying Proposition 3, we conclude:

Corollary 8. *The solution of a normalized system S of alternation depth $d > 1$ and with n iteration variables can be computed in time*

$$O\Big(h \cdot \Big(\frac{h \cdot n}{d-1} + 1\Big)^{\lceil d/2 \rceil} \cdot |S|\Big). \quad \square$$

Corollary 9. *A fixpoint expression e with alternation depth $d > 1$ and n occurrences of μ or ν can be evaluated in time*

$$O\Big(h \cdot \Big(\frac{h \cdot n}{d-1} + 1\Big)^{\lceil d/2 \rceil} \cdot |e|\Big). \quad \square$$

5. Discussion and conclusion

We presented a simple method for computing the solution of hierarchical system S of equations. The algorithm has the same or even slightly better estimation of its worst-case complexity than the one sketched in [12]. Additional benefit is gained from the new distinction between iteration variables and auxiliary ones.

In light of our exposition, the algorithmic idea of Long et al. in [12] can be interpreted as evaluating system S' (which we constructed by syntactical fixpoint removal) *without* explicitly constructing S'. Their hope is that usually fixpoints are reached after much fewer iterations than theoretically possible.

It seems, however, that this potential gain may as well be achieved by our method if we do not construct S' in advance which clearly may cause a tremendous waste in space and time but create an algorithmic description of it and apply some general demand-driven equation solver (e.g., the one from [10]) which builds up the system to be evaluated "by need".

A practical evaluation on real world benchmarks is highly desirable to find out whether the presented theoretical improvement also leads to practical improvements over existing conventional algorithms, e.g., the one of Cleaveland et al. [7].

Acknowledgement

Many thanks to André Arnold who patiently listened to rough sketches of the ideas and encouraged submission for publication.

References

[1] H.R. Andersen, Model checking and Boolean graphs, *Theoret. Comput. Sci.* **126** (1) (1994) 3-30.

[2] A. Arnold and P. Crubille, A linear time algorithm for solving fixpoint equations on transition systems, *Inform. Process. Lett.* **29** (1988) 57-66.

[3] O. Bernholtz, M.V. Vardi and P. Wolper, An automata-theoretic approach to branching-time model checking (extended abstract), in: *Proc. 6th Conf. on Computer Aided Verification*, Lecture Notes on Computer Science **818** (Springer, Berlin, 1994) 142-155.

[4] G. Bhat and R. Cleaveland, Efficient local model checking for fragments of the modal μ-calculus, in: *Proc. Tools and Algorithms for the Construction and Analysis of Systems*, Lecture Notes on Computer Science **1055** (Springer, Berlin, 1996) 107-126.

[5] J.R. Burch, E.M. Clarke and K.L. McMillan, Symbolic model checking 10^{20} states and beyond, *Inform. and Comput.* **98** (1992) 142-170.

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

[7] R. Cleaveland, M. Klein and B. Steffen, Faster model checking for the modal μ-calculus, in: *Proc. 4th Conf. on Computer Aided Verification*, Lecture Notes on Computer Science **663** (Springer, Berlin, 1992) 410-422.

[8] E.A. Emerson and C.-L. Lei, Efficient model checking in fragments of the propositional μ-calculus, in: *Proc. 1st IEEE Symp. on Logic in Computer Science* (1986) 267-278.

[9] E.A. Emerson, C.S. Jutla and A.P. Sistla, On model-checking for fragments of μ-calculus, in: *Proc. 5th Conf. on Computer Aided Verification*, Lecture Notes on Computer Science **697** (Springer, Berlin, 1993) 385-396.

[10] C. Fecht and H. Seidl, An even faster solver for general systems of equations, Tech. Rept. 96-11, Trier, 1996; also in: *Proc. 3rd Int. Static Analysis Symposium* (1996), to appear.

[11] D. Kozen, Results on the propositional μ-calculus, *Theoret. Comput. Sci.* **27** (1983) 333-354.

[12] D.E. Long, A. Browne, E.M. Clarke, S. Jha and W.R. Marrero, An improved algorithm for the evaluation of fixpoint expressions, in: *Proc. 6th Conf. on Computer Aided Verification*, Lecture Notes on Computer Science **818** (Springer, Berlin, 1994) 338-350.

[13] C. Stirling, Modal and temporal logics, in: S. Abramsky, D. Gabbay and T. Maibaum, eds., *Handbook of Logic in Computer Science* (1992) 477-563.

[14] M.Y. Vardi, A temporal fixpoint calculus. in: *Proc. 15th Internat. Conf. on Principles of Programming Languages* (1988) 250-259.

[15] M.Y. Vardi and P. Wolper, Automata-theoretic techniques for modal logics of programs, *J. Comput. System Sci.* **32** (1986) 183-221.