Efficient Local Correctness Checking

Kim Guldstrand Larsen
Aalborg University, Denmark *

Motivation

This paper deals with the problem of verifying the correctness of a finite-state parallel system presented in terms of a finite labeled transition system. More precisely, we consider *logical* as well as *behavioural* specifications and the associated correctness checking problems (model–checking and equivalence/preorder–checking).

A number of theoretical techniques and tools have been developed. Here we mention the algorithms presented in [EL86, CS91b, PT87, PS90] and the tools [LMV88, RRSV87, EES86, CPS88]. However, traditionally the techniques applied are *global*, in the sense that they prerequire the generation (and storage) of the complete transition system before verification. As the size of the the global transition systems may grow exponentially in the number of parallel components, the main limitations of these traditional tools has been a space problem. In the last few years there have been a growing interest in techniques that avoid this global preconstruction. Here we mention the work of [FM91, CS91a, GW91b, GW91a, And92]. In particular the work in [CS91a] and [And92] are closely related to ours.

However, existing work in this direction has been developed mainly for specific correctness problems. In this paper we provide an abstract and uniform description of an efficient and local technique, which we show applicable to a variety of model–checking and equivalence/preorder–checking problems. In fact, the abstract technique we present is the very heart of all tools of the TAV–system [JGZ89].

Our general technique is based on a notion of consistency of Boolean Equation Systems (section 1), in terms of which a number of correctness problems may be represented (section 2). We show briefly how consistency may be checked using a well–known global technique (section 3). In the paper we provide two *proof systems* for determining consistency with respect to a Boolean Equation System. We provide suitable soundness and completeness theorems for the proof systems, and indicate how to extract *local* consistency checking algorithms. The first proof system (section 4) captures the essence of a number of recently developed (and implemented) model–checking techniques for the modal mu–calculus [Lar90, SW89, Cle90, Win89]. However, these techniques yield exponential time worst case complexity. The second proof system (section 5) remedies this deficiency and yields a polynomial–time local consistency checking algorithm. In the conclusion we discuss how the presented local technique — which is based on the behavioural semantics of processes — may be combined with algebraic properties of processes.

* Address: Dep. of Math. and Comp. Sc., Aalborg University, Fredrik Bajersvej 7, 9220 Aalborg, Denmark. Telephone: +45 98158522. Email: kgl@iesd.auc.dk. This work is done in the context of ESPRIT Basic Research Action 7166, CONCUR2. Also the work has been supported in part by the Danish Natural Science Research Council through the DART project.

1 Boolean Equation Systems

In this section we shall present the notion of *Boolean Equation Systems*. As we shall see in the next section, many correctness problems encountered in the area of parallel and reactive systems may by represented and solved through the use of Boolean Equation Systems.

The basis of Boolean Equation Systems is that of a negation–free, propositional formula. Let V be a set of (propositional) variables. The set $\mathcal{L}_V$ of negation–free, propositional formulae over V is given by the following abstract syntax:

$$\phi \;::=\; x \mid \mathtt{tt} \mid \mathtt{ff} \mid \phi_1 \wedge \phi_2 \mid \phi_1 \vee \phi_2$$

where $x \in V$. We say that a formula ϕ is *simple* if it is in one of the forms $\mathtt{tt}$, $\mathtt{ff}$, $x_1 \wedge x_2$ or $x_1 \vee x_2$, where x_1 and x_2 are variables from V.

Semantically, we interpret formulae with respect to an *environment* $\rho : V \longrightarrow \mathrm{Bool}$ mapping variables to booleans ($\mathrm{Bool} = \{\mathtt{0}, \mathtt{1}\}$). More precisely, for ρ an environment and ϕ a formula we define the boolean value $[\![\phi]\!]\rho$ inductively on ϕ as follows:

$$\begin{aligned}
[\![x]\!]\rho &= \rho(x)\\
[\![\mathtt{tt}]\!]\rho &= \mathtt{1}\\
[\![\mathtt{ff}]\!]\rho &= \mathtt{0}\\
[\![\phi_1 \wedge \phi_2]\!]\rho &= \min\{[\![\phi]\!]\rho, [\![\phi_2]\!]\rho\}\\
[\![\phi_1 \vee \phi_2]\!]\rho &= \max\{[\![\phi]\!]\rho, [\![\phi_2]\!]\rho\}
\end{aligned}$$

Syntactically the desired semantics of variables of V is specified recursively through the use of an *equation system*, $E : V \longrightarrow \mathcal{L}_V$ over V. That is, E is a function which for each variable x gives the (recursive) definition, $E(x) \in \mathcal{L}_V$, for x. We shall write $x =_E \phi$ to indicate that $E(x) = \phi$. We are now able to give a succinct definition of a *Boolean Equation System*:

Definition 1. *A* Boolean Equation System *is a pair*

$$\mathcal{E} = \langle V, E\rangle$$

where V is a finite set of variables and E is an equation system over V. The Boolean Equation System $\langle V, E\rangle$ is said to be simple *in case $E(x)$ is a simple formula for any variable x.*

An equation system E specifies a semantic requirement to an environment ρ. In particular, for any variable x, $[\![x]\!]\rho$ must equal $[\![E(x)]\!]\rho$. If ρ has this property, we call it a *model* with respect to E. We identify on this basis two sets of variables:

Definition 2. *Let $\mathcal{E} = \langle V, E\rangle$ be a Boolean Equation System. A variable x is said to be* consistent *with respect to $\mathcal{E}$ if $\rho(x) = \mathtt{1}$ for some model ρ of $\mathcal{E}$. We denote by $C_{\mathcal{E}}$ the set of all consistent variables. If a variable x is not consistent we call it* inconsistent.
A variable x is said to be factual *with respect to $\mathcal{E}$ if $\rho(x) = \mathtt{1}$ whenever ρ is a model of $\mathcal{E}$. We denote by $S_{\mathcal{E}}$ the set of all factual variables.*

Given a Boolean Equation System $\mathcal{E} = \langle V, E\rangle$ we may define a functional $\mathcal{F}_\mathcal{E} : 2^V \longrightarrow 2^V$ by:

$$\mathcal{F}_\mathcal{E}(A) = \{x \in V \mid [\![E(x)]\!]\rho_A = \mathtt{1}\}$$

where $A \subseteq V$, and ρ_A is the characteristic function for A yielding $\mathtt{1}$ for any variable in A and $\mathtt{0}$ for variables outside A. As we only allow the use of negation–free formulae in an equation system it may be shown that $\mathcal{F}_\mathcal{E}$ is a monotonic function on the complete lattice of subsets of V ordered by set–inclusion. Using the standard fixed–point result due to Tarski [Tar55], this implies that $\mathcal{F}_\mathcal{E}$ has a maximal fixedpoint, $\nu\mathcal{F}_\mathcal{E}$, as well as a minimal fixedpoint, $\mu\mathcal{F}_\mathcal{E}$. It is routine to show that $\nu\mathcal{F}_\mathcal{E}$ coincides with the set of consistent variables $C_\mathcal{E}$ and that $\mu\mathcal{F}_\mathcal{E}$ coincides with the set of factual variables $S_\mathcal{E}$.

Example 1. Let $\mathcal{E} = \langle V, E\rangle$ be the Boolean Equation System, where $V = \{x_1, x_2, x_3, x_4\}$ and E is defined by the following four equations:

$$x_1 =_E x_1 \wedge x_2 \qquad x_3 =_E \mathtt{ff}$$
$$x_2 =_E x_3 \vee x_4 \qquad x_4 =_E \mathtt{tt}$$

It should then be obvious that $C_\mathcal{E} = \{x_1, x_2, x_4\}$ and $S_\mathcal{E} = \{x_2, x_4\}$. □

In logic a formula is called *consistent* provided one cannot infer contradictions from it. Our notion of consistency corresponds closely to this usage of the term: a variable x is consistent provided you cannot infer $\mathtt{0} = \mathtt{1}$ from the assumption that $x = \mathtt{1}$. Similarly, our use of the term *factuality* corresponds to the standard notion of theoremhood in logic. In the following sections we shall concentrate on methods for checking consistency. Methods for factuality checking may be obtained by straightforward dualisation.

In the analysis we shall state our complexity results in terms of the *size* of a Boolean Equation System $\mathcal{E} = \langle V, E\rangle$ defined as $|\mathcal{E}| = \sum_{x \in V} |E(x)|$, where the size of a formula ϕ is defined inductively as: $|\mathtt{tt}| = |\mathtt{ff}| = |x| = 1$ and $|\phi_1 \wedge \phi_2| = |\phi_1 \vee \phi_2| = |\phi_1| + |\phi_2|$. Also, we shall make some general assumptions about the representation of a Boolean Equation System. Firstly, we shall assume that variables are represented by natural numbers (which we in turn assume to be representable in constant amount of memory). Secondly, functions from finite subsets of natural numbers (e.g. E of a Boolean Equation System $\mathcal{E} = \langle V, E\rangle$) will be represented with efficient access to the value of one particular element in the domain (constant time as for "array" in many programming languages). Finally, note that any Boolean Equation System $\mathcal{E} = \langle V, E\rangle$ may be transformed into a Simple Boolean Equation System with only a linear blow–up: if $x =_E \phi_1 \vee \phi_2$ and ϕ_1 and ϕ_2 are compound formulae themselves simply add two new variables x_{ϕ_1} and x_{ϕ_2} and replace the above equation with the following three: $x =_E x_\phi \vee x_{\phi_2}$, $x_{\phi_1} =_E \phi_1$ and $x_{\phi_2} =_E \phi_2$. Repeating this procedure will eventually result in the desired Simple Boolean Equation System.

2 Representing Correctness Problems

We adopt the reactive view of parallel processes advocated in [Pnu85]; i.e. we model the behaviour of a process in terms of a *labelled transition system* describing its potential interaction with the environment. Having adopted this view, the *correctness* of a process may be formulated in a variety of ways: In the *process algebraic* framework [Mil80, Mil89, Hoa85, BK85, Bou85, BB87], a number of behavioural equivalences and preorders exists for comparing (concrete and abstract) processes. Alternatively, one may use formulae of Temporal and Modal Logic [BAPM83, Koz82] for specifying the desired behaviour of a process.

We claim that several of these notions of correctness may be represented as consistency (and factuality) problems of Boolean Equation Systems, thus allowing a single, uniform treatment. Due to lack of space we justify this claim by only a few illustrating examples. In particular, we show in this section how to represent bisimulation equivalence problems [Par81, Mil83] and simulation problems [Mil83, Lar87] between (finite–state) processes as Boolean Equation Systems. These two correctness problems are just a small sample of problems representable in terms of Boolean Equation Systems, and they have been selected because of their simplicity. Other problems which might have been presented include: $\frac{2}{3}$–bisimulation [LS91] (or ready bisimulation [Blo88]), m–nested simulation, refinement between modal transition systems [LT88], equation solving problems [Shi, Par89, LX90] and other synthesis problems. Also, model–checking problems as well as satisfiability problems with respect to the modal nu–calculus are representable as Boolean Equation Systems.

Definition 3. *A* labelled transition system *is a structure $\mathcal{P} = (S, A, \longrightarrow)$ where S is a set of states, A is a set of actions and $\longrightarrow \subseteq S \times A \times S$ is the transition relation. A labeled transition system $\mathcal{P}$ is said to be* finite *provided S and A (and hence $\longrightarrow$) are finite.*

The well–known notions of *simulation* and *bisimulation* [Par81, Mil83, Lar87] provide means of identifying processes based on their operational behaviour. Below we recall their formal definitions:

Definition 4. *Let $\mathcal{P} = (S, A, \longrightarrow)$ be a labelled transition system. Then a* simulation *R is a binary relation on S such that whenever $(P, Q) \in R$ and $a \in A$ then the following holds:*

– *Whenever $P \xrightarrow{a} P'$, then $Q \xrightarrow{a} Q'$ for some Q' with $(P', Q') \in R$,*

Q is said to simulate *P in case (P, Q) is contained in some simulation R. We write $P \preceq Q$ in this case. A binary relation R on S is a* bisimulation *in case both R and R^T are simulations* [2]*. P and Q are said to be bisimilar in case (P, Q) is contained in some bisimulation R. We write $P \sim Q$ in this case.*

We now provide the representation of simulation and bisimulation as Boolean Equation Systems:

Definition 5. *Let $\mathcal{P} = (S, A, \longrightarrow)$ be a finite labelled transition system. Then the Boolean Equation System $\mathcal{E}^{\preceq}_{\mathcal{P}} = \langle V, E\rangle$ is defined as $X_{P,Q} \in V$ whenever $P, Q \in S$, and*

$$X_{P,Q} =_E \bigwedge_{P \xrightarrow{a} P'} \left(\bigvee_{Q \xrightarrow{a} Q'} X_{P',Q'} \right)$$

[2] For R a binary relation the transposed relation R^T is defined as $R^T = \{(Q, P) | (P, Q) \in R\}$.

The Boolean Equation System $\mathcal{E}^{\sim}_{\mathcal{P}} = \langle V, E\rangle$ *is defined as* $Y_{P,Q} \in V$ *whenever* $P, Q \in S$, *with*

$$Y_{P,Q} =_E \bigwedge_{P \xrightarrow{a} P'} \left(\bigvee_{Q \xrightarrow{a} Q'} Y_{P',Q'} \right) \wedge \bigwedge_{Q \xrightarrow{a} Q'} \left(\bigvee_{P \xrightarrow{a} P'} Y_{P',Q'} \right)$$

The correctness of the above representations are stated in the following theorem:

Theorem 6. *Let* $\mathcal{P} = (S, A, \longrightarrow)$ *be a finite labelled transition system. Then* $X_{P,Q}$ *is consistent with respect to* $\mathcal{E}^{\precsim}_{\mathcal{P}}$ *if and only if* Q *simulates* P. *Also,* $Y_{P,Q}$ *is consistent with respect to* $\mathcal{E}^{\sim}_{\mathcal{P}}$ *if and only if* P *and* Q *are bisimilar.*

3 Global Correctness Checking

For a Boolean Equation System $\mathcal{E} = \langle V, E\rangle$ the set of *consistent* variables may be computed in a straightforward and well–known manner, which is applied in several existing tools (e.g. [LMV88, CPS88]): simply compute the following decreasing sequence of variable–sets:

$$V \supsetneq \mathcal{F}_{\mathcal{E}}(V) \supsetneq \mathcal{F}^2_{\mathcal{E}}(V) \supsetneq \cdots\cdots \mathcal{F}^n_{\mathcal{E}}(V) = \mathcal{F}^{n+1}_{\mathcal{E}}(V) \tag{1}$$

That is, starting with the set of all variables V, we simply apply the functional $\mathcal{F}_{\mathcal{E}}$ repeatedly until convergence is reached (in (1) this happens after $n+1$ iterations). As there are only finitely many variables, termination is guaranteed, and standard fixedpoint theory [Tar55] ensures that the set obtained at convergence is the maximal fixedpoint of $\mathcal{F}_{\mathcal{E}}$, i.e. the set of consistent variables.

As for complexity of this method, convergence is clearly obtained after at most $|V|$ iterations. In each iteration we must compute the semantic value of $E(x)$ for each variable x. Assuming that the access time for each variable is constant, this can be computed in time $\sum_{x\in V} |E(x)| = |\mathcal{E}|$. Hence, the complete worst case time complexity is $O(|V||\mathcal{E}|)$, which for Simple Boolean Equation Systems is the same as $O(|V|^2)$.

4 Local Correctness Checking

Using the global approach to consistency (or dually, factuality) checking, one is forced to consider all variables (in fact all variables are considered in each iteration). However, the initial problem might be concerned with the consistency (or factuality) of a *particular* variable, in which case the global technique seems to be an overkill. Instead we would want consistency of a given variable to be determined based on information of only a few (related) variables. For correctness problems in the world of parallel, reactive systems the global technique prerequires a total state–space construction with the familiar state–space explosion as a likely consequence. In contrast, we would prefer to settle the correctness problem of a parallel system in a manner that would minimize the construction and examination of its state–space.

In the following we shall present a *local* technique for consistency (and factuality) checking in a manner that exploits the Boolean Equation System in a minimal fashion.

Example 2. Consider the Simple Boolean Equation System given below:

$$x_1 = x_1 \wedge x_1 \qquad x_2 = x_1 \vee x_3 \qquad x_3 = \mathtt{ff}$$

Clearly, $C_{\mathcal{E}} = \{x_1, x_2\}$. However, as x_1 is completely independent of x_2 (and x_3) it should be possible to infer consistency of x_1 without any information of x_2 (and x_3). □

In Figure 1 we present the proof system $\mathcal{A}$ for inferring (relative) consistency of variables of a *simple* Boolean Equation System $\mathcal{E} = \langle V, E\rangle$. The statements of the proof system are of the form

$$\{x_1, \ldots, x_n\} \vdash_{\mathcal{E}} x \tag{2}$$

where $x_1, \ldots, x_n$ and x are variables of V. The statement in (2) may informally be interpreted as: the variable x is consistent under the assumption of consistency of $x_1, \ldots, x_n$. Most of the rules are obvious. However, note in rule $\mathbf{A}_3$ that the consistency of a variable x may be inferred from the consistency of its definition, under an assumption–set *augmented* with the variable itself. As consistency is defined using a *maximal* fixedpoint, this turns out to be a sound rule.

$$\mathbf{A}_1\ \frac{-}{\Gamma \vdash_{\mathcal{E}} x}\ x \in \Gamma \qquad\qquad \mathbf{A}_2\ \frac{-}{\Gamma \vdash_{\mathcal{E}} \mathtt{tt}}$$

$$\mathbf{A}_3\ \frac{\Gamma, x \vdash_{\mathcal{E}} \phi}{\Gamma \vdash_{\mathcal{E}} x}\ x =_{\mathcal{E}} \phi,\ x \notin \Gamma$$

$$\mathbf{A}_4\ \frac{\Gamma \vdash_{\mathcal{E}} x \quad \Gamma \vdash_{\mathcal{E}} y}{\Gamma \vdash_{\mathcal{E}} x \wedge y}$$

$$\mathbf{A}_5\ \frac{\Gamma \vdash_{\mathcal{E}} x}{\Gamma \vdash_{\mathcal{E}} x \vee y} \qquad\qquad \mathbf{A}_6\ \frac{\Gamma \vdash_{\mathcal{E}} y}{\Gamma \vdash_{\mathcal{E}} x \vee y}$$

Fig. 1. $\mathcal{A}$: Local Checking of Consistency

Formally, we may show the following soundness theorem:

Theorem 7. *Let* $\mathcal{E} = \langle V, E\rangle$ *be a Simple Boolean Equation System. Then*

$$\Gamma \vdash_{\mathcal{E}} x \ \Rightarrow\ \exists C. \left\{ \begin{array}{l} C \backslash \Gamma \subseteq \mathcal{F}_{\mathcal{E}}(C)\ \wedge \\ x \in C \end{array} \right\}$$

Proof By induction on the inference structure. Here we only consider the case when $\Gamma \vdash_{\mathcal{E}} x$ has been established using rule $\mathbf{A}_3$. That is, $\Gamma, x \vdash_{\mathcal{E}} \phi$ where $x =_{\mathcal{E}} \phi$. As $\mathcal{E}$ is simple, ϕ will be either a disjunction or a conjunction of variables. Assuming the latter — i.e. $\phi = x_1 \wedge x_2$ — then $\Gamma, x \vdash_{\mathcal{E}} \phi$ must have been inferred using $\mathbf{A}_4$ and thus $\Gamma, x \vdash_{\mathcal{E}} x_1$ and $\Gamma, x \vdash_{\mathcal{E}} x_2$. Appealing now to the Induction Hypothesis, we may conclude that $C_i \backslash (\Gamma \cup \{x\}) \subseteq \mathcal{F}_{\mathcal{E}}(C_i)$

and $x_i \in C_i$ for some C_i $(i = 1,2)$. Now let $C = C_1 \cup C_2 \cup \{x\}$, then clearly $x \in C$. Also $C \setminus \Gamma \subseteq \mathcal{F}_{\mathcal{E}}(C)$ due to monotonicity and definition of $\mathcal{F}_{\mathcal{E}}$. □

As an easy corollary we may infer that the inference system is sound with respect to consistency:

Corollary 8. *Let $\mathcal{E} = \langle V, E \rangle$ be a Simple Boolean Equation System. Whenever $\emptyset \vdash_{\mathcal{E}} x$ then $x \in C_{\mathcal{E}}$.*

Example 3. Reconsider the Simple Boolean Equation System from Example 2. Using the proof system $\mathcal{A}$, consistency of x_1 may be inferred as follows:

$$\mathbf{A}_3\,\dfrac{\mathbf{A}_4\,\dfrac{\mathbf{A}_1\,\dfrac{-}{\{x_1\} \vdash_{\mathcal{E}} x_1} \qquad \dfrac{-}{\{x_1\} \vdash_{\mathcal{E}} x_1}\,\mathbf{A}_1}{\{x_1\} \vdash_{\mathcal{E}} x_1 \wedge x_1}}{\emptyset \vdash_{\mathcal{E}} x_1}$$

Note, that the consistency of x_1 has been inferred in a local fashion without information about x_2 and x_3. □

The following theorem claims that Figure 1 constitutes a complete inference system for consistency.

Theorem 9. *Let $\mathcal{E} = \langle V, E \rangle$ be a Simple Boolean Equation System. Whenever $x \in C_{\mathcal{E}}$ then $\emptyset \vdash_{\mathcal{E}} x$.*

Proof Now let $\Gamma \prec \Omega$ whenever $\Omega \subset \Gamma$. Given that V only contains finitely many variables $\prec$ will be a well–founded ordering. Now, using well–founded induction on Γ we may show that $\Gamma \vdash_{\mathcal{E}} x$ whenever x is contained in some postfixed point relative to Γ. □

Using the inference rules of $\mathcal{A}$ in a goal–directed and backwards manner with possible backtracking (due to the choice between the or–rules $\mathbf{A}_5$ and $\mathbf{A}_6$) we clearly obtain a decision procedure for consistency checking (easily implemented in PROLOG). However, as we shall see the induced decision procedure has exponential worst–case time complexity and is thus — though clearly a local checking technique — inferior to the classical global technique of the previous section. We shall see in the next section how to remedy this deficiency.

5 Efficient Local Correctness Checking

The local checking technique of the previous section describes the essence of a number of recent techniques for modelchecking in the modal mu–calculus [Koz82], including the proof–system of [Lar90], the tableau system of [SW89, Cle90] (which has been incorporated into the Concurrency Workbench [CPS88]) and the rewrite system in [Win89]. However, in all cases the techniques have an exponential worst case time complexity as illustrated by the following Simple Boolean Equation System:

$$x_0 = x_1 \vee x_1, \quad x_1 = x_2 \vee x_2, \quad \cdots\cdots x_{n-1} = x_n \vee x_n, \quad x_n = \mathtt{ff} \tag{3}$$

Trying (and obviously failing) to demonstrate consistency of x_0 using the inference rules of figure 1 in a goal–directed manner with possible backtracking will lead to a computation with $2^{n+1} - 1$ recursive invocations as illustrated in figure 2 for the case $n = 2$. Here the superscripts indicate the order in which the invocations fails. The subscripts describe a sequence of rules that when applied to a node will yield the parent node.

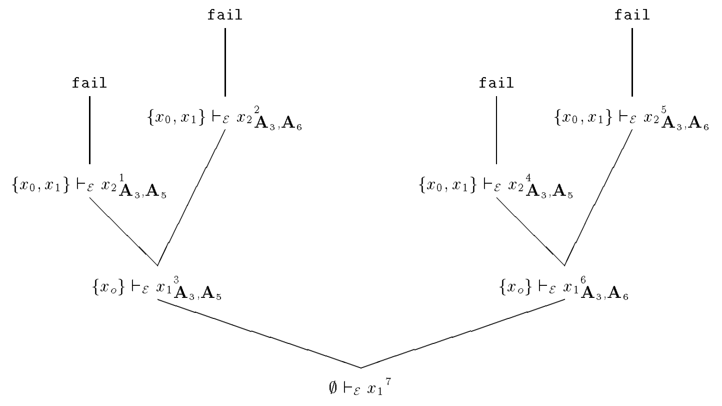

Fig. 2. Exponential Time Computation in $\mathcal{A}$.

Clearly, the source of the inefficiency is caused by the fact that *failing* attempts of establishing (relative) consistency are not remembered, and hence must necessarily result in recomputations when reencountered.

In figure 3 we present a proof system $\mathcal{B}$ for consistency checking in a manner which remembers and recalls previously discovered *in*consistency results. Given a Simple Boolean Equation System $\mathcal{E} = \langle V, E \rangle$ the proof system $\mathcal{B}$ permits the inference of statements of the form:

$$\langle x, B, N \rangle \leadsto_{\mathcal{E}} \langle b, C, M \rangle \tag{4}$$

where x is a variable of V; B, N, C and M are subsets of V, and b is a boolean value. In (4), $\langle x, B, N \rangle$ should be thought of as the problem given: is x consistent under the *assumption* that the variables in B are consistent and *knowing* that the variables in N are inconsistent? $\langle b, C, M \rangle$ then describes the answer to this question: the boolean b directly indicates the consistency of x with respect to B and N, whereas C and M are extensions of B and N containing results of consistency gathered during the process of answering the question $\langle x, B, N\rangle$ [3]. In particular, these extensions will be useful in "pruning" subsequent computations.

$$\mathbf{B}_1 \langle x, B, N\rangle \rightsquigarrow_{\mathcal{E}} \langle \mathtt{1}, B\cup\{x\}, N\rangle \qquad \text{when } x \in B \text{ or } x =_{\mathcal{E}} \mathtt{tt}$$

$$\mathbf{B}_2 \langle x, B, N\rangle \rightsquigarrow_{\mathcal{E}} \langle \mathtt{0}, B, N\cup\{x\}\rangle \qquad \text{when } x \in N \text{ or } x =_{\mathcal{E}} \mathtt{ff}$$

$$\mathbf{B}_3 \frac{\langle \phi, B\cup\{x\}, N\rangle \rightsquigarrow_{\mathcal{E}} \langle \mathtt{1}, C, M\rangle}{\langle x, B, N\rangle \rightsquigarrow_{\mathcal{E}} \langle \mathtt{1}, C, M\rangle} \text{ when } x \notin B\cup N \text{ and } x =_{\mathcal{E}} \phi$$

$$\mathbf{B}_4 \frac{\langle \phi, B\cup\{x\}, N\rangle \rightsquigarrow_{\mathcal{E}} \langle \mathtt{0}, C, M\rangle}{\langle x, B, N\rangle \rightsquigarrow_{\mathcal{E}} \langle \mathtt{0}, B, M\cup\{x\}\rangle} \text{ when } x \notin B\cup N \text{ and } x =_{\mathcal{E}} \phi$$

$$\mathbf{B}_5 \frac{\langle x_1, B, N\rangle \rightsquigarrow_{\mathcal{E}} \langle \mathtt{0}, C, M\rangle}{\langle x_1 \wedge x_2, B, N\rangle \rightsquigarrow_{\mathcal{E}} \langle \mathtt{0}, C, M\rangle}$$

$$\mathbf{B}_6 \frac{\langle x_1, B, N\rangle \rightsquigarrow_{\mathcal{E}} \langle \mathtt{1}, C, M\rangle \qquad \langle x_2, C, M\rangle \rightsquigarrow_{\mathcal{E}} \langle b, D, R\rangle}{\langle x_1 \wedge x_2, B, N\rangle \rightsquigarrow_{\mathcal{E}} \langle b, D, R\rangle}$$

$$\mathbf{B}_7 \frac{\langle x_1, B, N\rangle \rightsquigarrow_{\mathcal{E}} \langle \mathtt{1}, C, M\rangle}{\langle x_1 \vee x_2, B, N\rangle \rightsquigarrow_{\mathcal{E}} \langle \mathtt{1}, C, M\rangle}$$

$$\mathbf{B}_8 \frac{\langle x_1, B, N\rangle \rightsquigarrow_{\mathcal{E}} \langle \mathtt{0}, C, M\rangle \qquad \langle x_2, C, M\rangle \rightsquigarrow_{\mathcal{E}} \langle b, D, R\rangle}{\langle x_1 \vee x_2, B, N\rangle \rightsquigarrow_{\mathcal{E}} \langle b, D, R\rangle}$$

Fig. 3. $\mathcal{B}$: Efficient Local Checking of Consistency

For a given problem $\langle x, B, N\rangle$, the proof system $\mathcal{B}$ is using the set B as an assumption set similar to the use of assumptions in the previous proof system $\mathcal{A}$. In fact, the following formal relationship between the two proof systems may be shown:

$$\text{Whenever } \langle x, B, N\rangle \rightsquigarrow_{\mathcal{E}} \langle \mathtt{1}, C, M\rangle \text{ in } \mathcal{B} \text{ then } B \vdash_{\mathcal{E}} x \text{ in } \mathcal{A}.$$

We first give an informal description of the rules of $\mathcal{B}$:

$\mathbf{B}_1, \mathbf{B}_2$: Axioms for inferring immediate (relative) consistency/inconsistency of a variable x. The rule $\mathbf{B}_1$ corresponds closely to the rule $\mathbf{A}_1$ of $\mathcal{A}$.

$\mathbf{B}_3, \mathbf{B}_4$: In case the variable x is not included in any of the sets B and N the consistency of x is reduced to the consistency of its definition (ϕ). However, similar to the rule $\mathbf{A}_3$ of $\mathcal{A}$, we are allowed to use x itself as an assumption, during the consistency checking of ϕ. In case ϕ turns out consistent under the extended assumption set, we simply inherit the complete result obtained from ϕ as the result of x. Otherwise — i.e. if ϕ is found inconsistent — also x is inconsistent, reflected in the augmentation of the result set M. In this case, we are not able to use any information gathered in the set C as it may be based on the (erroneous) assumption of x being consistent.

$\mathbf{B}_5, \mathbf{B}_6, \mathbf{B}_7, \mathbf{B}_8$: The obvious rules for (simple) conjunctive and disjunctive formulae with information gathered being passed from the first conjunct (disjunct) to the subsequent processing of the second conjunct (disjunct).

[3] Those familiar with the terminology of Structured Operational Semantics [Plo81] may view the inference system $\mathcal{B}$ as a *natural* semantics in the sense that it models *full* computations.

Now reconsider the Simple Boolean Equation System of (3) again for $n = 2$. In figure 4 we use the new proof system $\mathcal{B}$ to infer the inconsistency of x_o. Here $\Gamma_i = \{x_0, \dots x_i\}$ and $\Omega_i = \{x_i, \dots, x_2\}$ for $i = 0, 1, 2$. In contrast to a goal–directed use of the previous proof system $\mathcal{A}$, we note that recomputation is totally avoided. In particular, we note that inconsistency of x_1 is only computed once; the second time x_1 is encountered $(*)$ the assumption-sets prevents a recomputation. As a consequence, as we shall state more precisely shortly, we avoid the exponential time complexity.

$$\mathbf{B}_4\,\dfrac{\mathbf{B}_8\,\dfrac{\mathbf{B}_4\,\dfrac{\mathbf{B}_8\,\dfrac{\vdots}{\langle x_2 \vee x_2, \Gamma_1, \emptyset\rangle \rightsquigarrow_{\mathcal{E}} \langle \mathtt{0}, \Gamma_1, \Omega_2\rangle}}{\langle x_1, \Gamma_0, \emptyset\rangle \rightsquigarrow_{\mathcal{E}} \langle \mathtt{0}, \Gamma_0, \Omega_1\rangle} \qquad \dfrac{-}{\langle x_1, \Gamma_0, \Omega_1\rangle \rightsquigarrow_{\mathcal{E}} \langle \mathtt{0}, \Gamma_0, \Omega_1\rangle}\,\mathbf{B}_2{}^{*}}{\langle x_1 \vee x_1, \Gamma_0, \emptyset\rangle \rightsquigarrow_{\mathcal{E}} \langle \mathtt{0}, \Gamma_0, \Omega_1\rangle}}{\langle x_0, \emptyset, \emptyset\rangle \rightsquigarrow_{\mathcal{E}} \langle \mathtt{0}, \emptyset, \Omega_0\rangle}$$

Fig. 4. Inconsistency in $\mathcal{B}$.

Formally, we have shown the following soundness result for $\mathcal{B}$:

Theorem 10. *Let $\mathcal{E} = \langle V, E\rangle$ be a Simple Boolean Equation System. Then, whenever $\langle x, B, N\rangle \rightsquigarrow_{\mathcal{E}} \langle b, C, M\rangle$ the following holds:*

i) $C\backslash B \subseteq \mathcal{F}_{\mathcal{E}}(C)$	*iv)* $N \subseteq M$
ii) $N \cap \nu\mathcal{F}_{\mathcal{E}} = \emptyset \Rightarrow M \cap \nu\mathcal{F}_{\mathcal{E}} = \emptyset$	*v)* $b = \mathtt{1} \Rightarrow x \in C$
iii) $B \subseteq C$	*vi)* $b = \mathtt{0} \Rightarrow x \in M$

As an easy corollary we may infer that $\mathcal{B}$ is sound with respect to consistency.

Corollary 11. *Let $\mathcal{E} = \langle V, E\rangle$ be a Simple Boolean Equation System.*

1. *Whenever $\langle x, \emptyset, \emptyset\rangle \rightsquigarrow_{\mathcal{E}} \langle \mathtt{1}, C, M\rangle$ then x is consistent, i.e. $x \in C_{\mathcal{E}}$.*
2. *Whenever $\langle x, \emptyset, \emptyset\rangle \rightsquigarrow_{\mathcal{E}} \langle \mathtt{0}, C, M\rangle$ then x is inconsistent, i.e. $x \notin C_{\mathcal{E}}$.*

In contrast to the proof system $\mathcal{A}$, where a statement may be inferred in a number of ways due to the open choice between $\mathbf{A}_5$ and $\mathbf{A}_6$ of $\mathcal{A}$, the imposed left–to–right sequentiality in $\mathcal{B}$ makes proofs of statements unique. Formally, we have the following:

Theorem 12. *Let $\mathcal{E} = \langle V, E\rangle$ be a Simple Boolean Equation System. Then the following holds:*

$$\forall \langle x, B, N\rangle \exists! \langle b, C, M\rangle.\ \langle x, B, N\rangle \rightsquigarrow_{\mathcal{E}} \langle b, C, M\rangle$$

where $\exists!$ indicates unique existence.

Proof Now let $(B, N) \prec (B', N')$ if either $B \supsetneq B'$ and $N = N'$ or $N \supsetneq N'$ (i.e. $\prec$ is a lexicographical ordering on pairs with the first component being minor). As V is finite, $\prec$ will be a well–founded ordering. Now, the above theorem may be proved using well–founded induction on (B, N) and appealing to Theorem 10. $\square$

In fact, not only does $\mathcal{B}$ provide a unique answer $\langle b, C, M\rangle$ for any given problem $\langle x, B, N\rangle$, the *proof* that $\mathcal{B}$ provides may also be shown to be unique.

From Theorem 10 and Theorem 12 it is clear that $\mathcal{B}$ is complete with respect to consistency in the following sense:

Corollary 13. *Let $\mathcal{E} = \langle V, E\rangle$ be a Simple Boolean Equation System.*

1. *Whenever x is consistent, i.e. $x \in C_{\mathcal{E}}$, then $\langle x, \emptyset, \emptyset\rangle \rightsquigarrow_{\mathcal{E}} \langle \mathtt{1}, C, M\rangle$ for some sets C and M,*
2. *Whenever x is inconsistent, i.e. $x \notin C_{\mathcal{E}}$, then $\langle x, \emptyset, \emptyset\rangle \rightsquigarrow_{\mathcal{E}} \langle \mathtt{0}, C, M\rangle$ for some sets C and M.*

For results of complexity, we need a minimum amount of terminology concerning *proof trees*. A proof tree T of $\mathcal{B}$ contains statements of $\mathcal{B}$ as nodes and is either an instance of one of the axioms $\mathbf{B}_1$ and $\mathbf{B}_2$ or is of the form:

$$T = \frac{T_1 \cdot \cdots \cdot T_n}{s} \tag{5}$$

where s is a statement of $\mathcal{B}$, and $T_1 \ldots T_n$ are themselves proof trees. s is the *conclusion* of T and should be obtainable from the conclusions of $T_1 \ldots T_n$ using one of the rules of $\mathcal{B}$. We now introduce a *total* ordering $\ll$ between the nodes of a proof tree (5) inductively as follows: s is the smallest node with respect to $\ll$; all nodes of T_i are smaller than the nodes of T_j whenever $i < j$. For the ordering among nodes of any T_i we appeal to the inductive construction. Clearly, $\ll$ is a total ordering with statements increasing in an 'up–or–right' direction.

The following lemma states the relation between $\prec$ and $\ll$:

Lemma 14. *Let T be a proof tree of $\mathcal{B}$ and let $s_i = (\langle x_i, B_i, N_i\rangle \rightsquigarrow_{\mathcal{E}} \langle b_i, C_i, M_i\rangle)$ for $i = 1, 2$ be nodes in T. Then $(B_2, N_2) \prec (B_1, N_1)$ whenever $s_1 \ll s_2$.*

As $\ll$ is total ordering among the nodes of a proof tree T, it follows that there can be no more nodes in T than the length of the longest decreasing $\prec$–sequence. Thus, for a given Simple Boolean Equation System $\mathcal{E} = \langle V, E\rangle$, any $\mathcal{B}$ proof tree will have no more than $|V|^2$ nodes. Under the assumption of constant access time for each variable of V, checking the side–condition of any of the rules of $\mathcal{B}$ may be done in constant time. Hence, using the rules in a goal–directed manner will yield an algorithm with $O(|V|^2)$ worst case running time.

Concluding Remarks

In this paper we have presented a proof system for efficient consistency–checking of variables of a (Simple) Boolean Equation System. We claim that several model–checking and equation/preorder–checking problems may be dealt with using the developed techniques. However, the techniques of this paper only leads to a model–checking method for pure *saftety* properties. Properties which can be expressed in the modal mu–calculus using only minimal fixedpoints may be dealt with by a simple dualization of the proof system $\mathcal{B}$. For properties requiring the used of both minimal and maximal fixedpoints our techniques may be extended to the modal mu–calculus of alternation depth one (i.e. any recursive subformula with alternating fixedpoint must be closed). To deal with mixed fixedpoints in this restricted sense, we simply consider lists of Boolean Equation Systems $[\mathcal{E}_1, \ldots, \mathcal{E}_n]$, where we alternate between consistency and factuality of variables and with variables of $\mathcal{E}_i$ only depending on variables of $\mathcal{E}_{i+1}, \ldots, \mathcal{E}_n$. Local model–checking for the *full* modal mu–calculus is contained in [Xin92], where an $O\left(m \times n \times \left(\frac{m \times n}{k}\right)^k\right)$ algorithm is presented. Here m is the size of the process P, n is the size of the formula F, and k is the alternation depth of F.

Applying the proof system $\mathcal{B}$ to $\mathcal{E}^{\sim}_{\mathcal{P}}$ and $\mathcal{E}^{\preceq}_{\mathcal{P}}$ we obtain local techniques for checking bisimulation equivalence and simulation preorder. The techniques will clearly be based entirely on the behavioural semantics of processes, and will in no way take account of their possible syntactic structure. However, it is possible to extend $\mathcal{B}$ so that certain algebraic properties of processes may be utilized. More precisely, for any consistency–preserving relation $\equiv$ [4] we may change the rule $\mathbf{B}_1$ as follows:

$$\mathbf{B}^{\equiv}_1 \langle x, B, N\rangle \rightsquigarrow_{\mathcal{E}} \langle \mathtt{1}, B \cup \{x\}, N\rangle \qquad \text{when } \exists y \in B.x \equiv y, \text{ or } x =_{\mathcal{E}} \mathtt{tt}$$

while maintaining a sound proof system with respect to consistency. Thus, it is sufficient that x is $\equiv$–related to some member of B in order to invoke the axiom (and hence avoid further computation). Now, we may represent various algebraic laws between processes as a relation $\equiv$ between variables of $\mathcal{E}^{\sim}_{\mathcal{P}}$ and $\mathcal{E}^{\preceq}_{\mathcal{P}}$. The fact that the parallel operator of CCS [Mil80, Mil89] is commutative and has *nil* as unit with respect to bisimulation may be represented as:

$$Y_{P|Q,R} \equiv Y_{Q|P,R}$$
$$Y_{P|nil,R} \equiv Y_{P,R}$$

Taking in this way account of commutativity (and similarly associativity) of the parallel operator will clearly yield a much improved algorithm in verification of systems with many identical components, as we may identify system states with the same number of components in any particular state. Also, the fact that *nil* is the smallest process with respect to $\preceq$ may be reflected as $X_{nil,P} \equiv \mathtt{tt}$ thus avoiding computation completely. However, checking the side–condition of $\mathbf{B}^{\equiv}_1$ may no longer necessarily be done in constant time, and it will depend highly on the combination of the particular example and the choice of $\equiv$ as to whether $\mathbf{B}^{\equiv}_1$ yields an improvement.

[4] $\equiv$ is consistency–preserving if $y \in C_{\mathcal{E}}$ and $x \equiv y$ implies $x \in C_{\mathcal{E}}$.

Finally, it is clear that the global technique (with the required state–space precomputation) cannot be used for processes with infinite state space. We are currently investigating the relationship between our proof system $\mathcal{B}$ and the tableau–technique of [HS91] for contextfree processes.

Acknowledgement

The author would like to thank Liu Xinxin for many inspiring discussions on the subject of local correctness checking. Also, we would like to thank Hans Hüttel for reading and commenting on drafts of this paper.

References

[And92] H.R. Andersen. Model checking and boolean graphs. *Lecture Notes In Computer Science, Springer Verlag*, 1992. In Proceedings of CAAP'92.

[BAPM83] M. Ben-Ari, A. Pnueli, and Z. Manna. The temporal logic of branching time. *Acta Informatica*, 20, 1983.

[BB87] T. Bolognesi and E. Brinksma. Introduction to the ISO specification language LOTOS. *Computer Networks and ISDN Systems*, 14, 1987.

[BK85] J.A. Bergstra and J.W. Klop. Algebra of communicating processes with abstraction. *Theoretical Computer Science*, 37:77–121, 1985.

[Blo88] Meyer Bloom, Istrail. bisimulation can't be traced. *Proceedings of Principles of Programming Languages*, 1988.

[Bou85] G. Boudol. Calcul de processus et verification. Technical Report 424, INRIA, 1985.

[Cle90] R. Cleaveland. Tableau–based model checkin in the propositional mu–calculus. *Acta Informatica*, 1990.

[CPS88] R. Cleaveland, J. Parrow, and B. Steffen. The concurrency workbench. University of Edinburgh, Scotland, 1988.

[CS91a] R. Cleaveland and B. Steffen. Computing behavioural relations, logically. *Lecture Notes In Computer Science, Springer Verlag*, 510, 1991.

[CS91b] R. Cleaveland and B. Steffen. A linear–time model–checking algorithm for ethe alternation–free modal mu–calculus. *Lecture Notes In Computer Science, Springer Verlag*, 1991. To appear in Proceedings of Third Workshop on Computer Aided Verification.

[EES86] E.M.Clarke, E.A. Emerson, and A.P. Sistla. Automatic verification of finite–state concurrent systems using temporal logic specifications. *TOPLAS*, 8(2), 1986.

[EL86] E.A. Emerson and C.L Lei. Efficient model checking in fragments of the propositional mu–calculus. *Proceedings of Logic in Computer Science*, 1986.

[FM91] J.C. Fernandez and L. Mounier. A tool set for deciding behavioural equivalence. *Lecture Notes In Computer Science, Springer Verlag*, 527, 1991.

[GW91a] P. Godefroid and P. Wolper. A partial approach to model–checking. *In Proceedings of Logic in Computer Science*, 1991.

[GW91b] P. Godefroid and P. Wolper. Using partial orders for the efficient verification of deadlock freedom and safety properties. *Lecture Notes In Computer Science, Springer Verlag*, 1991. To appear in Proceedings of Third workshop on Computer Aided Verification.

[Hoa85] C.A.R. Hoare. *Communicating Sequential Processes*. Prentice–Hall, 1985.

[HS91] H. Hüttel and C. Stirling. Actions speak louder than words: proving bisimilarity for context–free processes. *Proceedings of Logic in Computer Science*, 1991.

[JGZ89] K.G. Larsen J.C. Godskesen and M. Zeeberg. Tav — tools for automatic verification — users manual. Technical Report R 89-19, Department of Mathematics and Computer Science, Aalborg University, 1989. Presented at workshop on Automatic Methods for Finite State Systems, Grenoble, France, Juni 1989.

[Koz82] D. Kozen. Results on the propositional mu–calculus. *Lecture Notes In Computer Science, Springer Verlag*, 140, 1982. in Proc. of International Colloquium on Algorithms, Languages and Programming 1982.

[Lar87] K.G. Larsen. A context dependent bisimulation between processes. *Theoretical Computer Science*, 49, 1987.

[Lar90] K.G. Larsen. Proof systems for satisfiability in Hennessy–Milner logic with recursion. *Theoretical Computer Science*, 72, 1990.

[LMV88] V. Lecompte, E. Madelaine, and D. Vergamini. Auto: A verfication system for parallel and communicating processes. INRIA, Sophia–Antipolis, 1988.

[LS91] K.G. Larsen and A. Skou. Bisimulation through probabilistic testing. *Information and Computation*, 94(1), 1991.

[LT88] Kim G. Larsen and Bent Thomsen. A modal process logic. In *Proceeding on Logic in Computer Science*, 1988.

[LX90] K.G. Larsen and L. Xinxin. Equation solving using modal transition systems. In *Proceedings on Logic in Computer Science*, 1990.

[Mil80] R. Milner. *Calculus of Communicating Systems*, volume 92 of *Lecture Notes In Computer Science, Springer Verlag*. Springer Verlag, 1980.

[Mil83] R. Milner. Calculi for synchrony and asynchrony. *Theoretical Computer Science*, 25:267–310, 1983.

[Mil89] R. Milner. *Communication and Concurrency*. Prentice–Hall, 1989.

[Par81] D. Park. Concurrency and automata on infinite sequences. *Lecture Notes In Computer Science, Springer Verlag*, 104, 1981. Proceedings of 5th GI Conference.

[Par89] J. Parrow. Submodule construction as equation solving in CCS. *Theoretical Computer Science*, 68, 1989.

[Plo81] G. Plotkin. A structural approach to operational semantics. FN 19, DAIMI, Aarhus University, Denmark, 1981.

[Pnu85] A. Pnueli. Linear and branching structures in the semantics and logics of reactive systems. *Lecture Notes In Computer Science, Springer Verlag*, 194, 1985. Proceedings of 12th International Colloquium on Automata, Languages and Programming.

[PS90] P.C.Kanellakis and S.A.Smolka. Ccs expressions, finite state processes, and three problems of equivalence. *Information and Control*, 86, 1990.

[PT87] Paige and Tarjan. Three partition refinement algorithms. *SIAM Journal of Computing*, 16(6), 1987.

[RRSV87] J.L. Rixchier, C. Rodriguez, J. Sifakis, and J. Voiron. Xesar: a tool for protocol validation. users' guide. Technical report, LGI–IMAG, 1987.

[Shi] M.W. Shields. A note on the simple interface equation. Technical report, University of Kent at Canterbury.

[SW89] C. Stirling and D. Walker. Local model checking in the modal mu–calculus. *Lecture Notes In Computer Science, Springer Verlag*, 352, 1989. In Proc. of Tapsoft'89.

[Tar55] A. Tarski. A lattice–theoretical fixpoint theorem and its applications. *Pacific Journal of Math.*, 5, 1955.

[Win89] G. Winskel. Model checking the modal nu–calculus. *Lecture Notes In Computer Science, Springer Verlag*, 372, 1989. In Proceedings of International Colloquium on Algorithms, Languages and Programming 19'89.

[Xin92] Liu Xinxin. *Specification and Decomposition in Concurrency*. PhD thesis, Aalborg University, 1992. R 92–2005.