A Compositional Proof System for the Modal μ-calculus and CCS. [1]

Sergey Berezin
Dept. of Computer Science,
Carnegie Mellon University,
5000 Forbes Ave.
Pittsburgh, PA 15213
U.S.A.
Email: berez+@cs.cmu.edu

Dilian Gurov
Dept. of Computer Science,
University of Victoria,
P.O.Box 3055
Victoria, BC
CANADA V8W 3P6
Email: dgurov@csr.uvic.ca

We present a Compositional Proof System for the modal μ-calculus and a generalized version of the parallel composition in CCS [11, 12]. The proof system is designed for inferring global properties of a system from the local properties of its components. This allows for efficient verification of parallel processes by decomposing the task into smaller problems of verifying the parallel components separately. In particular, the system can be used to combine model checking [6] with theorem proving. Since parallel composition causes the largest blow-up in the number of states, this technique proposes an effective solution to the state space explosion problem. The Proof System is implemented in PVS theorem prover [13], and the proof of its soundness was thoroughly checked using PVS logic as a metalanguage. The proof strategy mechanism of PVS can be used to achieve some degree of automation in a proof search.

1 Introduction.

In this paper we present a Compositional Proof System for the modal μ-calculus and CCS [11, 12]. We use a (slightly modified version of) CCS as a model of concurrency. Many systems of parallel processes can be expressed as CCS processes, and then checked against specifications in the modal μ-calculus. Following Stirling [14], our proof system consists of two subsystems. The first one deals with model checking CCS processes without the parallel composition operator, i.e. it contains proof rules for sequents of the form $p \vdash \Phi$ ("process p satisfies a formula Φ"), and is described in detail in [8] for the more general process algebra of Value Passing CCS and a first order μ-calculus. The other subsystem, which is presented in this paper, is devoted to a parallel composition operator and is designed to prove sequents given by $\Phi || \Psi \vdash \Theta$ ("for any processes p and q satisfying Φ and Ψ respectively, the composite system $p||q$ satisfies Θ"). These two proof systems with an additional inference rule from [14]:

$$\frac{p \vdash \Phi \quad \Phi||\Psi \vdash \Theta \quad q \vdash \Psi}{p||q \vdash \Theta}\ (||)$$

result in a compositional proof system for CCS (now *with* parallel composition operator) and the modal μ-calculus. Both subprocesses in each parallel composition operator have associated formulas specifying their properties. Whenever we are to prove a property of a parallel composition, we first prove that the corresponding properties hold for each component, and then infer in the proof system that the global property of the composition also holds. This compositional step substantially simplifies the verification problem, since it avoids building the whole state space for the parallel composition in finite-state case. This state space grows exponentially in the number of processes involved, thus causing the *state explosion problem.* Thus, as a particular case, we propose a promising method of combining model checking with theorem proving, when the verification of the components is accomplished by model checking.

Our verification framework also supports a *compositional design* in the sense that one can work out specifications for all the parts of a complex system and prove by our method that if every component satisfies its specification, then the whole design is correct. After the implementation it is enough to verify each component separately. Moreover, one can change the actual implementation of some components without having to repeat the verification of the entire system as soon as the new implementation meets its local requirements.

Our compositional approach differs from many others [2, 3, 5, 7] in that it can handle the parallel composition operator in a purely compositional way, remains general for the full CCS and the full modal μ-calculus, and at the same time is feasible for automation. In [2, 3, 5] the parallel composition operator was eliminated basically by encoding one of the subprocesses into the formula. In the worst case this results in an exponential blow-up in the size of the formula, and the total complexity remains the same as for non-compositional model checking [6].

[1]This research was sponsored by the National Science Foundation under grant no. CCR-9217549.

The proof systems of Stirling [14] and Dam [7] are closer in spirit to our system. However, our approach offers two main advantages over these proof systems: all rules are truly compositional, and fixpoint formulas are handled using tags. The latter eliminates the need for having complicated global rules and termination conditions [7], and thus makes the whole system easier to machine-assist. Also, the proof system of Stirling is restricted to Hennessy-Milner logic, which is too weak for many practical purposes.

In [7, 14] the least compositional rules are the ones for dealing with τ-actions. The reason for this is that in the original CCS the action τ is an abstraction for an atomic internal communication. This action can be generated in different ways, each of which is compositional in itself. But the compositionality is lost as soon as we abstract away the actual communication taking place. Such an abstraction is essential in modelling large communicating systems, since the details of internal communications do not contribute to understanding the external behavior of a system, and only increase the complexity of its description. The situation is quite different when we are concerned with verification of concurrent systems. Certain information about the internal communications that can produce the τ-action helps us to recover the compositionality by treating each individual case separately. It is also of great value to know what actions a particular process can not perform a-priori, either because such actions do not appear in the alphabet of the process, or because they have been explicitly restricted. These considerations have lead us to the modification of CCS τ action into the set of all *neutral* actions. Each neutral action a either results from the synchronization of *input* and *output* actions $a?$ and $a!$ resp., or is an internal action of some subprocess. In other words, we distinguish τ actions that arise from different synchronizations. Also, we generalize the parallel composition operator to have *restriction sets* for each of the two processes: $p\,_{\Gamma}\|_{\Delta}\,q$. This extension is further explained in Section 2.

The only case which still remains not satisfactorily compositional is when a process can perform both an input (output) action $a?$ ($a!$) and the corresponding neutral action a. This case, however, would not arise at all if one follows the modelling discipline advocated by R. Milner in [11]. The parallel composition and restriction are introduced separately only in order to retain some useful theoretical properties, but are conceptually meant to go together. Therefore, the proof system presented here does not provide rules for this case at all, considering it as being the result of bad modelling.

The idea of tagging fixpoint formulas has been first suggested in [16]. The tagging approach allows us to keep track of what sequents have already occurred in the proof tree and conclude that they are hence true. So, there is no need in introducing global rules with complicated termination conditions as in [7]. The main difficulty in adopting this approach in our setting is how to justify semantically the proof rules for fixpoint formulas, i.e. how to choose an appropriate semantics for tagged formulas. The solution to this problem presented here might seem rather complicated, but is natural and sound. The resulting proof rules are syntactically easy to apply, which is certainly more important than the complexity of justifying them. In [7] this trade-off was compromised in favor of ease of theoretical investigations and logical traditions, but at the expense of ease of use and machine assistance.

Our proof system is implemented in PVS theorem prover [13]. The PVS specification language is used as a metalanguage to specify and prove the soundness of all the inference rules and axioms. The proof system is encoded as a set of theorems, which can be used as rewrite rules while a proof is in progress. Since PVS has a built-in model checker, both steps of the verification of finite-state systems, i.e. model checking the components and deriving the global property, can be done in a single framework. Also, PVS provides a powerful mechanism of writing proof strategies for automated proof search in our system.

The paper is organized as follows. Section 2 describes our version of CCS. Section 3 introduces the modal μ-calculus [9] (syntax and semantics), and provides some examples of useful properties. Section 4 describes the Compositional Proof System and shows an example of a proof in the proof system. In Section 5 we argue for the soundness of the proof system, in particular for the soundness of the fixed point rules. In Section 6 we discuss the issue of implementation in PVS and two examples that we verified. We conclude in Section 7.

2 The Process Algebra.

We use the standard CCS of R. Milner [11, 12], except that we change the parallel composition operator and the means of synchronization. The importance of this change will become clear in section 4, where we need it to simplify the compositional inference rules. Instead of actions $\{a, \ldots\}$, co-actions $\{\bar{a}, \ldots\}$ and the special action τ, we define *input* $\{a?, \ldots\}$, *output* $\{a!, \ldots\}$ and *neutral* $\{a, \ldots\}$ actions respectively. We will denote actions of arbitrary type by Greek letters $\gamma, \delta, \ldots$. Now two processes in a parallel composition may synchronize by input and output actions of the same name, yielding the corresponding neutral action (one might write this fact as $a? \cdot a! = a! \cdot a? = a$). In other words, we distinguish between τ-actions which are formed by different pairs of actions.

Our *parallel composition operator* also has a more general form in comparison with CCS: $p\,{}_\Gamma\|_\Delta\, q$ can be considered roughly as $(p \upharpoonright \Gamma)|(q \upharpoonright \Delta)$ in the original CCS, where Γ and Δ are sets of action symbols. This operator is taken from [2]. Thus, the abstract grammar of our CCS is the following:

$$p \ ::= \ \mathbf{0} \mid \mathsf{P} \mid \gamma.p \mid p_0 + p_1 \mid p_0\,{}_\Gamma\|_\Delta\, p_1 \mid p \upharpoonright \Lambda \mid p\{\Xi\}.$$

Here $\mathbf{0}$ is the *nil* process (called *inaction* in [11]), that can not perform any action, P is a *process identifier*, $\gamma.p$ is a *prefix* operator, $p+q$ is a non-deterministic choice, $\upharpoonright \Lambda$ and $\{\Xi\}$ are *restriction* and *relabelling*. Process identifiers are declared using an *identifier declaration* of the form

$$\mathsf{P} \equiv p.$$

We will denote the set of all CCS processes by $\mathcal{P}$. The operational semantics of our CCS is shown on figure 1.

$$\frac{p \xrightarrow{\gamma} q}{\mathsf{P} \xrightarrow{\gamma} q} \left(\begin{array}{c} \mathsf{P} \ \equiv \ p \\ \text{is declared} \end{array}\right) \qquad \frac{}{\gamma.p \xrightarrow{\gamma} p} \qquad \frac{p \xrightarrow{\gamma} p'}{p+q \xrightarrow{\gamma} p'} \qquad \frac{q \xrightarrow{\gamma} q'}{p+q \xrightarrow{\gamma} q'}$$

$$\frac{p \xrightarrow{\gamma} q}{p \upharpoonright \Lambda \xrightarrow{\gamma} q \upharpoonright \Lambda} \ (\gamma \in \Lambda) \qquad \frac{p \xrightarrow{\gamma} q}{p\{\Xi\} \xrightarrow{\delta} q\{\Xi\}} \ (\Xi(\gamma) = \delta) \qquad \frac{p \xrightarrow{a?} p' \quad q \xrightarrow{a!} q'}{p\,{}_\Gamma\|_\Delta\, q \xrightarrow{a} p'\,{}_\Gamma\|_\Delta\, q'} \left(\begin{array}{l} a? \in \Gamma, \\ a! \in \Delta \end{array}\right)$$

$$\frac{p \xrightarrow{\gamma} p'}{p\,{}_\Gamma\|_\Delta\, q \xrightarrow{\gamma} p'\,{}_\Gamma\|_\Delta\, q} \ (\gamma \in \Gamma) \qquad \frac{q \xrightarrow{\delta} q'}{p\,{}_\Gamma\|_\Delta\, q \xrightarrow{\delta} p\,{}_\Gamma\|_\Delta\, q'} \ (\delta \in \Delta) \qquad \frac{p \xrightarrow{a!} p' \quad q \xrightarrow{a?} q'}{p\,{}_\Gamma\|_\Delta\, q \xrightarrow{a} p'\,{}_\Gamma\|_\Delta\, q'} \left(\begin{array}{l} a! \in \Gamma, \\ a? \in \Delta \end{array}\right)$$

Figure 1: Operational semantics of the CCS.

As an example, consider these simple processes:

$$\begin{array}{l}
\mathsf{P} \equiv a.b!.\mathsf{P} \\
\mathsf{Q} \equiv b?.c.\mathsf{Q} \\
\mathsf{R} \equiv (\mathsf{P}\,{}_\Gamma\|_\Delta\, \mathsf{Q}) \upharpoonright \Lambda, \\
\quad \text{where } \Gamma = \{a, b!\},\ \Delta = \{b?, c\} \text{ and } \Lambda = \{a, b, c\}.
\end{array}$$

The process R is combined from the two processes P and Q, that perform asynchronous actions a and c and are forced to synchronize by $b?$ and $b!$, since $b?, b! \notin \Lambda$.

3 The Modal μ-Calculus.

3.1 Syntax.

Definition 1. The language of the *modal μ-calculus* [9] consists of the following alphabet:

- $P, Q, \ldots \in$ *Prop*, are propositional constant symbols; in particular, we assume the existence of two constants `true` and `false`;
- $X, Y, \ldots \in$ *Var*, are propositional variables;
- $\gamma, \delta, \ldots \in$ *Act* are action symbols.

We assume that the set of action symbols *Act* consists of *input* $\{a?, \ldots\}$, *output* $\{a!, \ldots\}$ and *neutral* $\{a, \ldots\}$ action symbols, in order to ensure compatibility with action symbols of CCS from the previous section.

Formulas are defined as follows:

1. P, where P is a propositional constant;

2. X, where X is a propositional variable;

3. $\Phi_1 \wedge \Phi_2$, $\Phi_1 \vee \Phi_2$, where Φ_1 and Φ_2 are formulas;

4. $\langle\gamma\rangle\,\Phi$, $[\gamma]\,\Phi$, where $\gamma \in \mathit{Act}$ and Φ is a formula;

5. $\mu X.\Phi$, $\nu X.\Phi$, where Φ is a formula.

Note that the absence of negation does not decrease the expressive power of the logic, since we can always rewrite formulas in a so-called *negation normal form*, where all negations are applied to atomic formulas only (i.e. to propositional constants and free variables), and then define new propositional constants with the complement interpretation: $\mathcal{L}(\overline{P}) = S - \mathcal{L}(P)$ (see the next subsection for semantics).

For example, some properties of processes P, Q and R from the previous section can be expressed as:

$$\begin{aligned}\Phi &\equiv \nu X.\,\langle a\rangle\,\langle b!\rangle\,X\\ \Psi &\equiv \nu X.\,\langle b?\rangle\,\langle c\rangle\,X\\ \Theta &\equiv \nu X.\,\langle a\rangle\,\mu Y.(\langle c\rangle\,X \vee \langle b\rangle\,Y)\end{aligned}$$

The formulas Φ and Ψ say that the corresponding pairs of actions can repeat infinitely often. The formula Θ says that after a and some finite number of b's the action c can be executed, and this pattern can repeat infinitely often.

Now we describe the formal semantics of the logic.

3.2 Semantics.

A model (Kripke structure) is a tuple

$$\mathcal{M} = (S, \rightarrow, \mathit{Act}, e, \mathcal{L}),$$

where S is a set of CCS processes, $\rightarrow \subseteq S \times \mathit{Act} \times S$ is the transition relation defined on figure 1 and projected on S, $e : \mathit{Var} \rightarrow 2^S$ is an interpretation of variables (environment), and $\mathcal{L} : \mathit{Prop} \rightarrow 2^S$ is an interpretation of propositional constant symbols. In order to be consistent with the intuitive semantics of CCS and [8], we will also assume that the set S is closed under the rules of figure 1 (i.e. *transition closed*). Otherwise we may have a situation where, say, the process $a.\mathbf{0}$ can not perform the action a in the model, if $\mathbf{0} \notin S$. Thus, it does not satisfy $\langle a\rangle\,\mathtt{true}$, which is counterintuitive. This restriction, however, is not necessary and all the results in this paper remain valid without it.

The semantic function $[\![.]\!]_{\mathcal{L}}e$ assigns *semantic sets* to μ-calculus formulas, and is defined inductively as follows:

$$\begin{aligned}
&[\![P]\!]_{\mathcal{L}}e = \mathcal{L}(P); \quad [\![X]\!]_{\mathcal{L}}e = e(X);\\
&\quad\text{in particular, } [\![\mathtt{true}]\!]_{\mathcal{L}}e = S, \quad [\![\mathtt{false}]\!]_{\mathcal{L}}e = \emptyset\\
&[\![\Phi_1 \wedge \Phi_2]\!]_{\mathcal{L}}e = [\![\Phi_1]\!]_{\mathcal{L}}e \cap [\![\Phi_2]\!]_{\mathcal{L}}e;\\
&[\![\Phi_1 \vee \Phi_2]\!]_{\mathcal{L}}e = [\![\Phi_1]\!]_{\mathcal{L}}e \cup [\![\Phi_2]\!]_{\mathcal{L}}e;\\
&[\![\langle\gamma\rangle\,\Phi]\!]_{\mathcal{L}}e = \{s \in S \mid \exists s' \in [\![\Phi]\!]_{\mathcal{L}}e : s \xrightarrow{\gamma} s'\};\\
&[\![[\gamma]\,\Phi]\!]_{\mathcal{L}}e = \{s \in S \mid \forall s' \in S : (s \xrightarrow{\gamma} s') \implies s' \in [\![\Phi]\!]_{\mathcal{L}}e\};\\
&[\![\nu X.\Phi]\!]_{\mathcal{L}}e = \bigcup\{S' \subseteq S \mid S' \subseteq [\![\Phi]\!]_{\mathcal{L}}e\,[X := S']\}\\
&[\![\mu X.\Phi]\!]_{\mathcal{L}}e = \bigcap\{S' \subseteq S \mid S' \supseteq [\![\Phi]\!]_{\mathcal{L}}e\,[X := S']\}
\end{aligned}$$

Here the *updated environment* $e\,[X := S']$ coincides with e on all variables, except maybe X, and

$$e\,[X := S']\,(X) = S'.$$

The semantics of the fixed points is well-defined by Tarski's Fixed-point Theorem [15], since all formulas are negation free. Thus, the semantic function is monotone on the interpretation of all free variables.

We will write $p \models_{\mathcal{M}} \Phi$ for $p \in [\![\Phi]\!]_{\mathcal{L}}e$, and will often omit the subscript $\mathcal{M}$ when this is unambiguous. We will also write $\models_{\mathcal{M}} \Phi$ to mean that $p \models_{\mathcal{M}} \Phi$ holds for *every* process $p \in S_{\mathcal{M}}$, and $\models \Phi$ to mean that $\models_{\mathcal{M}} \Phi$ holds for all models, or is *valid* or *generally true*.

3.3 Extensions.

To make formulas shorter, we will use *compound actions* (denoted by $\alpha, \beta, \ldots$) in the modal operators. Compound actions are formed from the ordinary actions from *Act* using the (finitary or infinitary) union operator: $\alpha \cup \beta$, with the semantics of a non-deterministic choice. More precisely, the compound actions may be viewed as sets of actions, where

$$\xrightarrow{\alpha} =_{df} \bigcup_{\gamma \in \alpha} \xrightarrow{\gamma}$$

Thus, the meaning of the modalities for such compound actions is the following:

$$[\alpha]\,\Phi \equiv \bigwedge_{\gamma \in \alpha} [\gamma]\,\Phi, \quad \langle\alpha\rangle\,\Phi \equiv \bigvee_{\gamma \in \alpha} \langle\gamma\rangle\,\Phi.$$

The classical CCS τ-action can be expressed now as the set of all neutral actions occurring in a process. Thus, we are able to employ both the convenience of hiding irrelevant details at the level of process description, and the high degree of compositionality in verification.

4 The Compositional Proof System.

In the sequel we fix a model $\mathcal{M} = (\mathcal{P}, \rightarrow, Act, e, \mathcal{L})$, where $\mathcal{P}$ is the set of all CCS processes. We choose the most general model, since the results described in this section remain valid for all practical submodels used in verification.

Definition 2. A *sequent* is an expression of the form $\Phi\,{}_\Gamma\|_\Delta\,\Psi \models_\Lambda \Theta$, where Φ, Ψ and Θ are formulas, and Γ, Δ and $\Lambda \subseteq Act$ are sets of action symbols.

The meaning of sequents for μ-calculus formulas can be expressed as follows:

$$\Phi\,{}_\Gamma\|_\Delta\,\Psi \models_\Lambda \Theta \iff$$

$$\forall p, q.(p \restriction \Gamma \models \Phi \text{ and } q \restriction \Delta \models \Psi \implies (p\,{}_\Gamma\|_\Delta\,q) \restriction \Lambda \models \Theta).$$

We apply the following scheme for proving the correctness of composite systems of the form $(p\,{}_\Gamma\|_\Delta\,q) \restriction \Lambda$: assume that we have already proven that $p \restriction \Gamma \models \Phi$ and $q \restriction \Delta \models \Psi$ for some formulas Φ and Ψ. To prove that $(p\,{}_\Gamma\|_\Delta\,q) \restriction \Lambda \models \Theta$ for a formula Θ it is sufficient to show that $\Phi\,{}_\Gamma\|_\Delta\,\Psi \models_\Lambda \Theta$ is valid. In other words, we can introduce an inference rule:

$$\frac{p \restriction \Gamma \vdash \Phi \quad \Phi\,{}_\Gamma\|_\Delta\,\Psi \vdash_\Lambda \Theta \quad q \restriction \Delta \vdash \Psi}{(p\,{}_\Gamma\|_\Delta\,q) \restriction \Lambda \vdash \Theta}\ (\|)$$

This inference rule was inspired by a similar rule of C. Stirling in [14].

In this paper we elaborate on the proof system for sequents of type $\Phi\,{}_\Gamma\|_\Delta\,\Psi \vdash_\Lambda \Theta$. For details on the proof system for $p \vdash \Phi$, where p is a sequential CCS term, the reader is referred to [8].

In our proof system we handle fixed points by assigning *tags* [16, 1] to the fixed point operators. Intuitively (although simplified), tags store the information that some particular sequents have already occurred below in the proof tree, assuming that the tree grows up from the goal to axioms. The current sequent is included in the tag of a fixed point formula when this formula gets unfolded. If the same sequent appears later in the proof, it is considered proved. This way of reasoning works for greatest fixed points on the right hand side and for least fixed points on the left hand side of the '$\vdash$' sign. In practice, when unfolding a fixed point formula, it is not necessary to include the whole sequent into the tag. It is sufficient to store only the two other formulas of the sequent. Thus, formally, tags are sets of pairs of formulas, associated with fixed point operators.

We extend the syntax of formulas by tags L in the fixed point operators as follows:

- $\mu X\{L\}\Phi$, $\nu X\{L\}\Phi$, where L is a finite set of pairs of formulas. I.e., $L = \{(\Psi_1, \Psi_2), \ldots\}$.

We will write fixed points with empty tags in the standard μ-calculus syntax, e.g. $\mu X.\Phi$ instead of $\mu X\{\emptyset\}\Phi$, and will not distinguish between them.

For technical reasons, to simplify the proof of soundness, we have developed a special semantics for sequents with tagged formulas, so that every rule in the proof system is locally sound, including the fixed point rules. A standard way to prove the local soundness of the fixed point rules is to use the *reduction Lemma* 4 [16, 1] (see the next section). In order to apply this lemma here, the semantics of the sequent $\Phi\,{}_{\Gamma}\|_{\Delta}\,\Psi \models_{\Lambda} \nu X\{L\}\Theta$ must be of the form $U \subseteq V$, where V is the (extended) semantic set of $\nu X\{L\}\Theta$, and U is some semantic set corresponding to the pair (Φ, Ψ). To apply the reduction lemma to the least fixed points on the left hand side (e.g. for $\mu X\{L\}\Phi\,{}_{\Gamma}\|_{\Delta}\,\Psi \vdash_{\Lambda} \Theta$), we need to rewrite the semantics of the sequent into an equivalent form: $U' \subseteq V'$, where U' is now the semantics of the least fixed point formula $\mu X\{L\}\Phi$, and V' is a semantic set for the pair (Ψ, Θ), possibly defined differently from the one for (Φ, Ψ) above.

Before introducing the new semantics of sequents, define the *extended semantics* of tagged formulas. Assume given two functions f_μ and f_ν that map pairs of formulas into subsets of $\mathcal{P}$ (e.g. $f_\mu(\Phi,\Psi) \subseteq \mathcal{P}$). Then the definition of the extended semantic function $[\![.]\!]^{(f_\mu,f_\nu)}_{\mathcal{L}}e$ coincides with the one of $[\![.]\!]_{\mathcal{L}}e$ from Section 3 on all the operators except the fixed points:

$$[\![\nu X\{L\}\Phi]\!]^{(f_\mu,f_\nu)}_{\mathcal{L}}e = \bigcup\{S' \subseteq \mathcal{P} \mid S' \subseteq [\![\vee L]\!]^{(f_\mu,f_\nu)}_{\mathcal{L}}e \cup [\![\Phi]\!]^{(f_\mu,f_\nu)}_{\mathcal{L}}e\,[X := S']\}$$
$$[\![\mu X\{L\}\Phi]\!]^{(f_\mu,f_\nu)}_{\mathcal{L}}e = \bigcap\{S' \subseteq \mathcal{P} \mid S' \supseteq [\![\wedge L]\!]^{(f_\mu,f_\nu)}_{\mathcal{L}}e \cap [\![\Phi]\!]^{(f_\mu,f_\nu)}_{\mathcal{L}}e\,[X := S']\}$$

where

$$[\![\vee L]\!]^{(f_\mu,f_\nu)}_{\mathcal{L}}e = \bigcup_{(\Phi,\Psi)\in L} f_\nu(\Phi,\Psi) \quad\text{and}\quad [\![\wedge L]\!]^{(f_\mu,f_\nu)}_{\mathcal{L}}e = \bigcap_{(\Phi,\Psi)\in L} f_\mu(\Phi,\Psi).$$

In particular,

$$[\![\vee \emptyset]\!]^{(f_\mu,f_\nu)}_{\mathcal{L}}e = \emptyset \qquad\qquad [\![\wedge \emptyset]\!]^{(f_\mu,f_\nu)}_{\mathcal{L}}e = \mathcal{P}.$$

Notice, that if all tags in a formula are empty, then its extended semantics coincides with the semantics defined in Section 3.

We need to provide suitable functions that assign semantic sets to pairs of formulas, as we discussed above. We call such functions *composition* and *left/right division operations.* They are introduced in Definition 4, using an additional operation of Γ-*closure* and similar operations for sets of CCS processes from Definition 3. The Γ-closure adds to the set of CCS processes all the processes that are not of the form $(p \restriction \Gamma)$ for this particular Γ. Lemma 1 allows to rewrite the semantics of a sequent in different representations (like $U \subseteq V$ and $U' \subseteq V'$ above).

Definition 3. Let A, B and C be subsets of $\mathcal{P}$. Define

- $(A)^\Gamma =_{df} \{q \mid (\exists p : q = (p \restriction \Gamma)) \implies q \in A\}$
- $(A\,{}_{\Gamma}\|_{\Delta}\,B) \restriction \Lambda =_{df} \{(p\,{}_{\Gamma}\|_{\Delta}\,q) \restriction \Lambda \mid (p \restriction \Gamma) \in A \text{ and } (q \restriction \Delta) \in B\}$
- $C/^{\mathit{left}}_{\Gamma,\Delta,\Lambda}B =_{df} (\{(p \restriction \Gamma) \mid \text{ for all } (q \restriction \Delta) \in B :\ (p\,{}_{\Gamma}\|_{\Delta}\,q) \restriction \Lambda \in C\})^\Gamma$
- $C/^{\mathit{right}}_{\Gamma,\Delta,\Lambda}A =_{df} (\{(q \restriction \Delta) \mid \text{ for all } (p \restriction \Gamma) \in A :\ (p\,{}_{\Gamma}\|_{\Delta}\,q) \restriction \Lambda \in C\})^\Delta$

Lemma 1. *For $A, B, C \subseteq \mathcal{P}$ the following holds:*

$$(A\,{}_{\Gamma}\|_{\Delta}\,B) \restriction \Lambda \subseteq C \iff A \subseteq C/^{\mathit{left}}_{\Gamma,\Delta,\Lambda}B \iff B \subseteq C/^{\mathit{right}}_{\Gamma,\Delta,\Lambda}A$$

Definition 4. Let Γ, Δ and Λ be subsets of Act. Then define

$$\begin{aligned}
\varepsilon(\Phi,\Psi) &=_{df} \emptyset \\
\mathsf{quotl}(\Phi,\Psi) &=_{df} [\![\Phi]\!]^{(\varepsilon,\mathsf{par})}_{\mathcal{L}}e/^{\mathit{left}}_{\Gamma,\Delta,\Lambda}[\![\Psi]\!]^{(\mathsf{quotr},\varepsilon)}_{\mathcal{L}}e \\
\mathsf{quotr}(\Phi,\Psi) &=_{df} [\![\Phi]\!]^{(\varepsilon,\mathsf{par})}_{\mathcal{L}}e/^{\mathit{right}}_{\Gamma,\Delta,\Lambda}[\![\Psi]\!]^{(\mathsf{quotl},\varepsilon)}_{\mathcal{L}}e \\
\mathsf{par}(\Phi,\Psi) &=_{df} ([\![\Phi]\!]^{(\mathsf{quotl},\varepsilon)}_{\mathcal{L}}e\,{}_{\Gamma}\|_{\Delta}\,[\![\Psi]\!]^{(\mathsf{quotr},\varepsilon)}_{\mathcal{L}}e) \restriction \Lambda,
\end{aligned}$$

It can be shown that this mutual recursion is well-defined. Note, that the functions quotl, quotr and par also depend on Γ, Δ and Λ, although we do not include these parameters for the sake of readability. The semantics of the sequent $\Phi\,{}_\Gamma\|_\Delta\,\Psi \models_\Lambda \Theta$ for tagged formulas Φ, Ψ and Θ is defined as follows (with the same Γ, Δ and Λ in par):

$$\Phi\,{}_\Gamma\|_\Delta\,\Psi \models_\Lambda \Theta \iff \mathsf{par}(\Phi,\Psi) \subseteq [\![\Theta]\!]_{\mathcal{L}}^{(\varepsilon,\mathsf{par})} e.$$

Lemma 2. *Assume that all ν-subformulas in Φ and Ψ and all μ-subformulas of Θ have empty tags. Then*

$$\Phi\,{}_\Gamma\|_\Delta\,\Psi \models_\Lambda \Theta \iff (\forall p,q : (p \restriction \Gamma) \in [\![\Phi]\!]_{\mathcal{L}}^{(\mathsf{quotl},\varepsilon)} e \text{ and } (q \restriction \Delta) \in [\![\Psi]\!]_{\mathcal{L}}^{(\mathsf{quotr},\varepsilon)} e$$

$$\implies (p\,{}_\Gamma\|_\Delta\, q) \restriction \Lambda \in [\![\Theta]\!]_{\mathcal{L}}^{(\varepsilon,\mathsf{par})} e)$$

We will define a sound approximation $\Phi\,{}_\Gamma\|_\Delta\,\Psi \vdash_\Lambda \Theta$, for which we can build a proof system. In this proof system all the proof rules preserve the conditions of Lemma 2. Therefore, if we start with formulas with empty tags, then all the sequents produced during the proof will satisfy these conditions.

The Compositional Proof System consists of axioms (fig. 2) and inference rules (fig. 3, 4 and 5). We say that a sequent $\Phi\,{}_\Gamma\|_\Delta\,\Psi \vdash_\Lambda \Theta$ is valid if there is a derivation of this sequent in the proof system.

We will not show all the rules in this paper; we provide only the rules dealing with the leftmost and the rightmost formulas (labelled, e.g. by $(l\,[.]\,1)$ and $(r\,[.])$). The corresponding rules for the middle formula are symmetrical with those for the left formula (referred to as, e.g. $(l\,[.]\,2)$).

Note, that we could not have most of the axioms and modal inference rules in such a simple form as they are if we had used τ-actions instead of neutral actions. For τ-actions, for example, the rule $[\tau 1]$ (figure 5) would have to have as many premises as there are synchronizable pairs of actions in Γ and Δ, or the formulas Φ and Ψ would have certain restrictions on all the other actions. This is inconvenient and unnecessary, since in our system we can represent the action τ by the set of all neutral actions a that arise from the synchronous execution of $a?$ and $a!$. In addition, we can also easily prove properties for only those synchronizations we are interested in.

An Example Proof. We will show here a short example proof for the processes P, Q, R and their specifications Φ, Ψ and Θ described in Sections 2 and 3.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{(\textbf{axiom})}{\Phi\,{}_\Gamma\|_\Delta\,\Psi \vdash_\Lambda \Theta_1}}{\Phi\,{}_\Gamma\|_\Delta\,\langle c\rangle\,\Psi \vdash_\Lambda \langle c\rangle\,\Theta_1}\,(l\,\langle.\rangle\,2)}{\Phi\,{}_\Gamma\|_\Delta\,\langle c\rangle\,\Psi \vdash_\Lambda \mu Y.(\langle c\rangle\,\Theta_1 \vee \langle b\rangle\,Y)}\,(r\mu,\ r\vee)}{\langle b!\rangle\,\Phi\,{}_\Gamma\|_\Delta\,\langle b?\rangle\,\langle c\rangle\,\Psi \vdash_\Lambda \langle b\rangle\,\mu Y.(\langle c\rangle\,\Theta_1 \vee \langle b\rangle\,Y)}\,(\langle\tau 2\rangle)}{\langle b!\rangle\,\Phi\,{}_\Gamma\|_\Delta\,\langle b?\rangle\,\langle c\rangle\,\Psi \vdash_\Lambda \langle c\rangle\,\Theta_1 \vee \langle b\rangle\,\mu Y.(\langle c\rangle\,\Theta_1 \vee \langle b\rangle\,Y)}\,(r\vee)}{\langle b!\rangle\,\Phi\,{}_\Gamma\|_\Delta\,\langle b?\rangle\,\langle c\rangle\,\Psi \vdash_\Lambda \mu Y.(\langle c\rangle\,\Theta_1 \vee \langle b\rangle\,Y)}\,(r\mu)}{\langle a\rangle\,\langle b!\rangle\,\Phi\,{}_\Gamma\|_\Delta\,\langle b?\rangle\,\langle c\rangle\,\Psi \vdash_\Lambda \langle a\rangle\,\mu Y.(\langle c\rangle\,\Theta_1 \vee \langle b\rangle\,Y)}\,(l\,\langle.\rangle\,1)}{\Phi\,{}_\Gamma\|_\Delta\,\Psi \vdash_\Lambda \Theta}\,(r\nu,\ l\nu 1,\ l\nu 2)$$

Where

$$\Theta_1 \equiv \nu X\{(\Phi,\Psi)\}\,\langle a\rangle\,\mu Y.(\langle c\rangle\,X \vee \langle b\rangle\,Y).$$

5 Soundness.

Soundness of the proof system described above can be stated as the following theorem:

Theorem 3. (Soundness) *Assume that all ν-subformulas in Φ and Ψ and all μ-subformulas of Θ have empty tags. Then*

$$\Phi\,{}_\Gamma\|_\Delta\,\Psi \vdash_\Lambda \Theta \implies \Phi\,{}_\Gamma\|_\Delta\,\Psi \models_\Lambda \Theta.$$

$$\mathtt{false}\,{}_\Gamma\|_\Delta\,\Psi \vdash_\Lambda \Theta \qquad \Phi\,{}_\Gamma\|_\Delta\,\mathtt{false} \vdash_\Lambda \Theta \qquad \Phi\,{}_\Gamma\|_\Delta\,\Psi \vdash_\Lambda \mathtt{true}$$

$$\begin{array}{ccc} \mu X\{L\}\Phi\,{}_\Gamma\|_\Delta\,\Psi \vdash_\Lambda \Theta & \Phi\,{}_\Gamma\|_\Delta\,\mu X\{L\}\Psi \vdash_\Lambda \Theta & \Phi\,{}_\Gamma\|_\Delta\,\Psi \vdash_\Lambda \nu X\{L\}\Theta \\ \text{if } (\Psi', \Theta') \in L & \text{if } (\Phi', \Theta') \in L & \text{if } (\Phi', \Psi') \in L \end{array}$$

where $\Phi' \trianglelefteq \Phi$, $\Psi' \trianglelefteq \Psi$ and $\Theta' \trianglelefteq \Theta$ in the last 3 axioms.

$$\Phi\,{}_\Gamma\|_\Delta\,\Psi \vdash_\Lambda [\alpha]\,\Theta$$

where for all $\gamma \in \alpha$: $(\gamma \notin \Lambda)$ or $(\gamma \notin \Gamma \cup \Delta$ and if $\gamma = a$ is neutral, then $\{(a?,\, a!),\, (a!,\, a?)\} \cap \Gamma \times \Delta = \emptyset)$

$$[\alpha]\mathtt{false}\,{}_\Gamma\|_\Delta\,\Psi \vdash_\Lambda [\alpha]\,\Theta\,(\alpha \cap \Delta = \emptyset) \quad \text{(where } \forall a \in \alpha.\ \{(a?,\, a!),\, (a!,\, a?)\} \cap \Gamma \times \Delta = \emptyset)$$

$$\langle\alpha\rangle\,\Phi\,{}_\Gamma\|_\Delta\,\Psi \vdash_\Lambda \Theta\,(\alpha \cap \Gamma = \emptyset) \qquad \Phi\,{}_\Gamma\|_\Delta\,\langle\beta\rangle\,\Phi \vdash_\Lambda \Theta\,(\beta \cap \Delta = \emptyset)$$

$$[a?]\mathtt{false}\,{}_\Gamma\|_\Delta\,\Psi \vdash_\Lambda [a]\,\Theta \qquad \Phi\,{}_\Gamma\|_\Delta\,[a!]\mathtt{false} \vdash_\Lambda [a]\,\Theta$$

where $(a! \notin \Gamma$ or $a? \notin \Delta)$ and $a \notin \Gamma \cup \Delta$ in the last 2 axioms.

Figure 2: Compositional Proof System: Axioms. The (syntactical) relation on formulas $\Phi \trianglelefteq \Psi$ means that the formulas have exactly the same structure except tags (that is, their untagged versions are the same), and all tags of Φ are subsets of the corresponding tags of Ψ.

$$(lw1)\,\frac{\Phi\,{}_\Gamma\|_\Delta\,\Psi \vdash_\Lambda \Theta}{\Omega\,{}_\Gamma\|_\Delta\,\Psi \vdash_\Lambda \Theta}\text{(where } \Phi \text{ and } \Omega \text{ have empty tags and } \models \Omega \longrightarrow \Phi)$$

$$(rw)\,\frac{\Phi\,{}_\Gamma\|_\Delta\,\Psi \vdash_\Lambda \Theta}{\Phi\,{}_\Gamma\|_\Delta\,\Psi \vdash_\Lambda \Omega}\text{(where } \Theta \text{ and } \Omega \text{ have empty tags and } \models \Theta \longrightarrow \Omega)$$

$$(l \wedge 1)\,\frac{\Phi\,{}_\Gamma\|_\Delta\,\Psi \vdash_\Lambda \Theta}{\Phi \wedge \Omega\,{}_\Gamma\|_\Delta\,\Psi \vdash_\Lambda \Theta} \qquad (l \vee 1)\,\frac{\Phi\,{}_\Gamma\|_\Delta\,\Psi \vdash_\Lambda \Theta \quad \Omega\,{}_\Gamma\|_\Delta\,\Psi \vdash_\Lambda \Theta}{\Phi \vee \Omega\,{}_\Gamma\|_\Delta\,\Psi \vdash_\Lambda \Theta}$$

$$(r\vee)\,\frac{\Phi\,{}_\Gamma\|_\Delta\,\Psi \vdash_\Lambda \Theta}{\Phi\,{}_\Gamma\|_\Delta\,\Psi \vdash_\Lambda \Theta \vee \Omega} \qquad (r\wedge)\,\frac{\Phi\,{}_\Gamma\|_\Delta\,\Psi \vdash_\Lambda \Theta \quad \Phi\,{}_\Gamma\|_\Delta\,\Psi \vdash_\Lambda \Omega}{\Phi\,{}_\Gamma\|_\Delta\,\Psi \vdash_\Lambda \Theta \wedge \Omega}$$

Figure 3: Compositional Proof System: Propositional Inference Rules.

$$(l\mu 1)\,\frac{\Phi[X/\mu X\{L \cup \{(\Psi,\Theta)\}\}\Phi]\,{}_\Gamma\|_\Delta\,\Psi \vdash_\Lambda \Theta}{\mu X\{L\}\Phi\,{}_\Gamma\|_\Delta\,\Psi \vdash_\Lambda \Theta} \qquad (r\nu)\,\frac{\Phi\,{}_\Gamma\|_\Delta\,\Psi \vdash_\Lambda \Theta[X/\nu X\{L \cup \{(\Phi,\Psi)\}\}\Theta]}{\Phi\,{}_\Gamma\|_\Delta\,\Psi \vdash_\Lambda \nu X\{L\}\Theta}$$

$$(l\nu 1)\,\frac{\Phi[X/\nu X\{L\}\Phi]\,{}_\Gamma\|_\Delta\,\Psi \vdash_\Lambda \Theta}{\nu X\{L\}\Phi\,{}_\Gamma\|_\Delta\,\Psi \vdash_\Lambda \Theta} \qquad (r\mu)\,\frac{\Phi\,{}_\Gamma\|_\Delta\,\Psi \vdash_\Lambda \Theta[X/\mu X\{L\}\Theta]}{\Phi\,{}_\Gamma\|_\Delta\,\Psi \vdash_\Lambda \mu X\{L\}\Theta}$$

Figure 4: Compositional Proof System: Fixed Point Inference Rules. The notation $\Phi\,[X/\Psi]$ denotes the substitution of Ψ for X in Φ.

$$([\tau 1])\,\frac{\Phi\,{}_\Gamma\|_\Delta\,\Psi \vdash_\Lambda \Theta}{[a?]\Phi\,{}_\Gamma\|_\Delta\,[a!]\Psi \vdash_\Lambda [a]\Theta}\begin{array}{l}(a \notin \Gamma \cup \Delta \text{ and}\\ (a!, a?) \notin \Gamma \times \Delta)\end{array} \qquad (\langle\tau 1\rangle)\,\frac{\Phi\,{}_\Gamma\|_\Delta\,\Psi \vdash_\Lambda \Theta}{\langle a?\rangle\Phi\,{}_\Gamma\|_\Delta\,\langle a!\rangle\Psi \vdash_\Lambda \langle a\rangle\Theta}(a \in \Lambda)$$

$$(l\,\langle . \rangle\,1)\,\frac{\Phi\,{}_\Gamma\|_\Delta\,\Psi \vdash_\Lambda \Theta}{\langle\alpha\rangle\Phi\,{}_\Gamma\|_\Delta\,\Psi \vdash_\Lambda \langle\alpha\rangle\Theta}(\alpha \subseteq \Lambda) \quad (l\,[.]\,1)\,\frac{\Phi\,{}_\Gamma\|_\Delta\,\Psi \vdash_\Lambda \Theta}{[\alpha]\Phi\,{}_\Gamma\|_\Delta\,\Psi \vdash_\Lambda [\alpha]\Theta}(\alpha \cap \Delta = \emptyset)$$

$$(lw\,[.]\,1)\,\frac{[\alpha]\mathtt{false}\wedge\Phi\,{}_\Gamma\|_\Delta\,\Psi \vdash_\Lambda [\beta]\Theta}{[\alpha]\mathtt{false}\wedge\Phi\,{}_\Gamma\|_\Delta\,\Psi \vdash_\Lambda [\alpha\cup\beta]\Theta}(\alpha \cap \Delta = \emptyset)$$

$$(r\,[.])\,\frac{\Phi\,{}_\Gamma\|_\Delta\,\Psi'\wedge[\beta]\Psi \vdash_\Lambda \Theta \qquad \Phi'\wedge[\alpha]\Phi\,{}_\Gamma\|_\Delta\,\Psi \vdash_\Lambda \Theta}{\Phi'\wedge[\alpha]\Phi\,{}_\Gamma\|_\Delta\,\Psi'\wedge[\beta]\Psi \vdash_\Lambda [\alpha\cap\beta]\Theta}$$

where $\forall a \in \alpha \cup \beta.\, \{(a?, a!), (a!, a?)\} \cap \Gamma \times \Delta = \emptyset$ in the last 3 rules.

Figure 5: Compositional Proof System: Modal Inference Rules.

Before sketching the proof we state the following lemma.

Lemma 4. (The reduction lemma [16, 1]). *Let D be a set and $f : 2^D \to 2^D$ be monotone with respect to $\subseteq$. Denote operators of the least and the greatest fixed points of f as $\mu x.f(x)$ and $\nu x.f(x)$ respectively. Then*

(i) $U \subseteq \nu x.f(x) \iff U \subseteq f(\nu x.(U \cup f(x)))$

(ii) $U \supseteq \mu x.f(x) \iff U \supseteq f(\mu x.(U \cap f(x)))$

Proof. (Of Theorem 3. Sketch).

We show soundness of the proof system by showing that all axioms and rules are individually sound (i.e. axioms are valid and rules preserve validity). For most of the rules the proof is a straightforward but tedious case analysis, using Lemma 2. The soundness of the fixed point rules $(l\mu 1)$, $(l\mu 2)$ and $(r\nu)$ follows directly from Lemmas 1 and 4. Fixed point axioms are valid because of the monotonicity of composition and division operators (composition is monotone on both arguments, division is antimonotone on the first argument and monotone on the second), and the following relations:

$$L \subseteq L' \implies [\![\vee L]\!]^{(f_\mu, f_\nu)} e \subseteq [\![\vee L']\!]^{(f_\mu, f_\nu)} e \text{ and } [\![\wedge L]\!]^{(f_\mu, f_\nu)} e \supseteq [\![\wedge L']\!]^{(f_\mu, f_\nu)} e$$

$$U \subseteq V \implies \mu x.(U \cap f(x)) \subseteq \mu x.(V \cap f(x)) \text{ and } \nu x.(U \cup f(x)) \subseteq \nu x.(V \cup f(x))$$

for a monotone f as in Lemma 4. □

The proof of soundness was completely checked using the theorem prover PVS [13]. All the inference rules are encoded as theorems and can be used as rewrite rules when a proof is in progress. Presently we are working on the completeness of the system. We expect that the system is complete for "canonical" processes (processes that have at most one of $a?$, $a!$ or a in their action alphabets for all actions). However, the proof may be very tedious, due to the automation-oriented form of the rules.

6 Implementation in PVS.

The Compositional Proof System is implemented in PVS theorem prover [13]. The main objectives of this implementation were to check the soundness of the system and to try out some relatively small proofs in the system. The PVS was chosen as an implementation framework because it has a built-in model checker. So, both steps of the verification of finite-state systems, i.e. model checking the components and deriving the global property, can be done in PVS. We verified two examples using the system: (1) Alternating Bit Protocol (ABP) [4, 6] and (2) Milner's Scheduler [12].

The ABP example consists of three parallel processes Send, Medium and Receive, combined together by two parallel composition operators:

$$\mathsf{ABP} \equiv ((\mathsf{Send}\ {}_{\Gamma_S}\|_{\Gamma_M}\ \mathsf{Medium})\ {}_\Delta\|_{\Gamma_R}\ \mathsf{Receive}) \restriction \Lambda$$

with appropriate restriction sets. Each individual process including the intermediate

$$(\mathsf{Send}\ {}_{\Gamma_S}\|_{\Gamma_M}\ \mathsf{Medium})$$

has its own specification. The specifications for the 'atomic' processes, i.e. Send, Medium and Receive, were directly model checked using SMV [10]. The specifications of compound processes (i.e. obtained by parallel composition) were derived from the components in the proof system.

The example of the Milner's scheduler is more involved and includes induction on the number of parallel processes. There are only two very simple 'atomic' processes: an *arbiter* p and a *short wire* sw. A scheduler with n arbiters is defined as

$$\mathsf{S}_n \equiv (\mathsf{B}_n\ {}_{\Gamma_n}\|_{\Delta_n}\ \mathsf{sw}\{\Xi_n\}) \upharpoonright \Lambda_n$$

and the *body* B_n is recursively defined by:

$$\begin{aligned} \mathsf{B}_1 &\equiv (\mathsf{p}\{\Xi_1\}) \upharpoonright \Gamma_1 \\ \mathsf{B}_{i+1} &\equiv (\mathsf{B}_i\ {}_{\Gamma_i}\|_{\Omega_{i+1}}\ \mathsf{p}\{\Xi_{i+1}\}) \upharpoonright \Gamma_{i+1} \end{aligned}$$

The relabelling Ξ_i is used to rename input and output actions so that they would not cause any confusion among different copies of p.

The verification was done by induction on the number of arbiters:

- Specifications for p and sw were model checked;
- Assuming proved $\mathsf{B}_n \models \Phi_n$ and $\mathsf{sw}\{\Xi_n\} \models \Psi_n$, the sequent $\Phi_n\ {}_{\Gamma_n}\|_{\Delta_n}\ \Psi_n \vdash_{\Lambda_n} \Theta_n$ was derived with n as a parameter;
- Also assuming $\mathsf{p}\{\Xi_n\} \models \phi_n$, the sequent $\Phi_n\ {}_{\Gamma_n}\|_{\Omega_{n+1}}\ \phi_{n+1} \vdash_{\Gamma_{n+1}} \Phi_{n+1}$ was proved.

After model checking of $\mathsf{B}_1 \models \Phi_1$ the verification was complete. Thus, we showed that for arbitrary n, $\mathsf{S}_n \models \Theta_n$ by induction on the number of parallel components.

The hardest part here was to find the right invariant Φ_n and to prove the sequent

$$\Phi_n\ {}_{\Gamma_n}\|_{\Omega_{n+1}}\ \phi_{n+1} \vdash_{\Gamma_{n+1}} \Phi_{n+1}.$$

The chart below shows the proof size in steps for several sequents:

Max Formula length	nesting depth	altern. depth	# proof steps (direct)	# proof steps (simplified)
15	3	2	23	23
17	5	2	80	80
23	3	2	$4.65 \cdot 10^8$	814

The column "# proof steps (simplified)" refers to the number of steps without repetitions of identical subproofs. As it can be easily seen, the naive proof tree ("direct" column) contains a lot of repetitions in the last proof (the induction step for the Milner's Scheduler). The blowup was caused by multiple instances of bound variables in fixed point operators, since the unfolding of fixed points produced several identical copies of subformulas. Thus, it is much more practical to think about a *proof DAG* (Directed Acyclic Graph) rather than a proof tree. Unfortunately, PVS does not allow to detect identical subgoals dynamically. Therefore, this proof system needs to be implemented in a special purpose theorem prover designed specifically for this system.

7 Conclusion.

We have presented a compositional proof system for the modal μ-calculus and a (more general version of a) parallel composition operator of CCS. The proof system allows us to decompose a verification task into simpler tasks for each parallel component. For example, in the finite state case, if we are to verify that a process term of the form $(P\ {}_{\Gamma}\|_{\Delta}\ Q) \upharpoonright \Lambda$ has a property Θ, we can reduce this task to showing that $P \upharpoonright \Gamma$ satisfies Φ and $Q \upharpoonright \Delta$ satisfies Ψ for some suitably chosen Φ and Ψ for which we can derive $\Phi\ {}_{\Gamma}\|_{\Delta}\ \Psi \vdash_{\Lambda} \Theta$ in our proof system. This way of compositional reasoning significantly reduces the state explosion problem arising in the direct model checking method [6]. In general, it is much easier to model check two properties of two components and prove a sequent $\Phi\,_\Gamma\|_\Delta\,\Psi \vdash_\Lambda \Theta$, than to model check the same property Θ for the result of the parallel composition directly. The reason is that in the finite-state case the parallel composition operator often causes an exponential blow-up of the number of states, and one may easily obtain an intractable size in a very simple example. In contrast, in our approach we would have to explore only several relatively small state spaces, and when formulas are not too long (which is often the case), produce tolerable overhead by deriving the global property, which results in computationally simpler and faster verification. Similar reasons work in the infinite-state case, except that we have to compare a different notion of complexity rather than the number of states.

Another significant advantage of the approach is that it supports a compositional design in the following sense. Suppose, we are to design a complex system consisting of dozens (if not hundreds or thousands) of parallel components. What we have to do first is to specify every component in some higher level specification language, and then make sure that if every specification is met, then the whole design will be correct. Our compositional proof system can naturally assist in solving this problem even before the actual implementation has started, and one may save significant amount of effort in case the specifications contain a subtle but crucial error. Moreover, after the implementation there is no need to verify the entire system. Instead, it is enough to prove the correctness of each of the components separately, which is a much simpler task.

There are many open problems in the area. To mention only the most important ones, we do not know if the proof system is complete in general or for any particular class of CCS processes. We are currently working on the completeness of the system for some sort of "canonical" processes. Another open question is the decidability of $\Phi\,_\Gamma\|_\Delta\,\Psi \models_\Lambda \Theta$. A positive answer would make a compositional model checking problem fully automatic and possibly tractable for virtually any size and complexity of finite-state systems.

In future we plan to implement this system more efficiently in a special purpose theorem prover and provide a better input language for writing specifications of parallel systems and their properties. We are also going to try this approach on industrial-size hardware/software designs.

Acknowledgments. We would like to thank Edmund Clarke, Bruce Kapron, Jürgen Dingel, Will Marrero and Marius Minea for careful reading and many useful comments. We are also grateful to Henrik Andersen for inviting the first author in January, 1995 to Danish Technical University, where this project was initiated.

References

[1] Henrik R. Andersen. On model checking infinite-state systems. In Nerode and Matiyasevich, editors, *LFCS'94: Logic at St. Petersburg. Symposium on Logical Foundations of Computer Science, St. Petersburg, Russia, July 11-14*, volume 813 of *Lecture Notes in Computer Science*. Springer-Verlag, 1994.

[2] Henrik R. Andersen. Partial model checking (extended abstract). Technical Report ID-TR: 1994-148, Department of Computer Science, Technical University of Denmark, October 1994. Accepted for LICS'95.

[3] Henrik R. Andersen, Colin Stirling, and Glynn Winskel. A compositional proof system for the modal μ-calculus. In *Proceedings, Ninth Annual IEEE Symposium on Logic in Computer Science*, pages 144–153, Paris, France, 4–7 July 1994. IEEE Computer Society Press. Also as BRICS Report RS-94-34.

[4] K.A. Bartlet, R.A. Scantlebury, and P.T. Wilkinson. A note on reliable full-duplex transmission over half-duplex links. *Commun. ACM*, 12(5):260–261, 1969.

[5] S.A. Berezine. Model checking in μ-calculus for distributed systems. In *Specification, verification, and net models of concurrent systems*. Institute of Informatics Systems, Novosibirsk, Russia, 1994.

[6] E. M. Clarke, E. A. Emerson, and A. P. Sistla. Automatic verification of finite-state concurrent systems using temporal logic specifications. *ACM Transactions on Programming Languages and Systems*, 8(2):244–263, April 1986.

[7] M. Dam. Compositional proof systems for model checking infinite state processes. In *Proceedings of CONCUR'95*, volume 962 of *Lecture Notes in Computer Science*, pages 12–26. Springer-Verlag, 1995.

[8] Dilian Gurov, Sergey Berezin, and Bruce M. Kapron. A modal mu-calculus and a proof system for value passing processes. In *INFINITY Workshop, Pisa, Italy*, volume 6 of *Electronic Notes in Theoretical Computer Science*, August 1996.

[9] D. Kozen. Results on the propositional mu-calculus. *Theoretical Computer Science*, 27:333–354, December 1983.

[10] K. L. McMillan. *Symbolic Model Checking: An Approach to the State Explosion Problem.* PhD thesis, Carnegie Mellon University, 1992.

[11] R. Milner. Calculi for synchrony and asynchrony. *Theoretical Computer Science*, 25(3):267–310, 1983.

[12] Robin Milner. *Communication and Concurrency.* Prentice-Hall, 1989.

[13] S. Owre, N. Shankar, and J. M. Rushby. *User Guide for the PVS Specification and Verification System.* Computer Science Laboratory, SRI International, Menlo Park, CA, February 1993. A new edition for PVS Version 2 is expected in late 1996.

[14] C. Stirling. Modal logics for communicating systems. *Theoretical Computer Science*, 49:311–348, July 1987.

[15] A. Tarski. A lattice-theoretic fixpoint theorem and its applications. *Pacific Journal of Mathematics*, 5:285–309, 1955.

[16] Glynn Winskel. A note on model checking the modal nu-calculus. In G. Ausiello, M. Dezani-Ciancaglini, and S. Ronchi Della Rocca, editors, *Proceedings of ICALP*, volume 372 of *Lecture Notes in Computer Science*, pages 761–772. Springer-Verlag, 1989.