ISSN 0105-8517

A Note on Model Checking the Modal ν-calculus†

Glynn Winskel

DAIMI PB – 279
June 1990
Revised Version

COMPUTER SCIENCE DEPARTMENT
AARHUS UNIVERSITY
Ny Munkegade, Building 540
DK-8000 Aarhus C, Denmark

† Revised version to appear in Theoretical Computer Science.

A note on model checking the modal ν-calculus

(Revised)

Glynn Winskel
Department of Computer Science
Aarhus University
Ny Munkegade, Bldg. 540
DK-8000 Aarhus, Denmark

This note presents a straightforward algorithm for checking whether or not a state of a labelled transition system satisfies an assertion in the modal ν-calculus and μ-calculus. The algorithm improves on those of Larsen, and Stirling and Walker in that its presentation lays bare the mechanism behind "local model checking", and leads to a streamlined proof of the correctness and termination of the model-checking algorithm.

Introduction

Labelled transition systems are used in modelling parallel processes. In the work on Milner's CCS a small modal logic called Hennessy-Milner logic has proved important in theoretical studies (see ch.10 of [6]). The expressiveness of Hennessy-Milner logic is too weak from a practical point of view however, and extending its assertions by recursive definitions is a simple way to increase its expressiveness dramatically [2]. This is one reason for a renewed interest in the modal μ-calculus of Pratt and Kozen, with minimum fixed points [7], [5] and its dual the modal ν-calculus, with maximum fixed points.

This work arose through trying to understand the work of Larsen [4], Stirling and Walker [8] on methods for deciding whether or not a state in a finite labelled transition system satisfies an assertion in the modal μ-calculus and ν-calculus. This enterprise has been described as "local model checking" by Stirling and Walker because their algorithm, and those of Larsen, take advantage of the fact that the goal is to establish whether or not a *particular* state satisfies an assertion. This is in contrast to the work of Emerson and Lei [2]. Emerson and Lei's treatment of recursive assertions is based on the observation that in a finite model the repeated unfoldings of a recursive definition must denote a stationary value from some finite stage onwards; the process of finding this stationary value is insensitive to the particular state of concern.

In his paper [4], Larsen showed how, in a sense, running his proof systems backwards provides a method to confirm a process satisfies a recursive modal assertion. His restrictions are quite severe however. There are no negations in his language of assertions and his method works only when assertions use purely minimum fixed points or purely maximum fixed points and nesting of them is certainly disallowed. This restricts the applicability of Larsen's work. The recent work of Walker and Stirling put this to rights. By including negation, one kind of fixed point operator, minimum or maximum, is definable in terms of the other. Their assertions include Hennessy-Milner logic with negations and maximum fixed points. Although equivalent to the modal μ-calculus, they really work with a modal ν-calculus. With respect to a finite model, they have a tableau method to decide whether or not a particular process satisfies an assertion. They have no restrictions on the kind of assertions they can check. A variant of the tableau method has been implemented by Cleaveland as part of the Edinburgh-Sussex Concurrency Workbench [3].

This note presents a very straightforward model-checking algorithm. The algorithm simplifies those of Larsen in [4], and Stirling and Walker in [8] and is accompanied by new proofs of soundness and completeness. A key improvement is that the effect of the assumption lists in the proof systems of Larsen, and Stirling and Walker is achieved by labelling recursions in assertions by sets of states. The model-checking algorithm can then be presented more simply as a reduction relation on correctness assertions; the reduction can be coded directly as a recursive function to decide the truth of an assertion at a process. This presentation lays bare the mechanism which guarantees termination of the algorithm and streamlines the proof of its correctness.

1 Maximum fixed points

We start with a special case of Tarski's theorem [9].

Theorem 1 *(Tarski)*

Let E be a set. Let $\phi : \mathcal{P}(E) \rightarrow \mathcal{P}(E)$ be a monotonic function i.e.

$$S \subseteq S' \Rightarrow \phi(S) \subseteq \phi(S')$$

for all $S, S' \in P(E)$. Then ϕ has a minimum fixed point $\mu S.\phi(S)$ and a maximum fixed point $\nu S.\phi(S)$ given by

$$\mu S.\phi(S) = \bigcap\{S' \subseteq E \mid \phi(S') \subseteq S'\}$$
$$\nu S.\phi(S) = \bigcup\{S' \subseteq E \mid S' \subseteq \phi(S')\}.$$

Say a subset $S' \subseteq E$ is a prefixed point of ϕ if

$$\phi(S') \subseteq S'$$

and a postfixed point of ϕ if

$$S' \subseteq \phi(S').$$

Then $\mu S.\phi(S)$ is the least prefixed point and $\nu S.\phi(S)$ is the greatest postfixed point.

Thinking of E as a set of states the extension of an assertion on states will be a subset of E. The importance of Tarski's theorem is that it shows that certain recursively defined assertions are sensible and gives techniques for reasoning about them. For example, the proof rules often given for reasoning about maximum fixed points have the form

$$\nu S.\phi(S) \subseteq \phi(\nu S.\phi(S)) \qquad \frac{S' \subseteq \phi(S')}{S' \subseteq \nu S.\phi(S)}$$

expressing that $\nu S.\phi(S)$ is the greatest postfixed point. One problem with these proof rules is that they are not backwards sound; it may well be that $S' \subseteq \nu S.\phi(S)$ without $S' \subseteq \phi(S')$

Backwards sound proof rules can be based on the following, a key fact on which the model checker is based.

Lemma 2 *(reduction lemma)*

Let ϕ be a monotonic function on $\mathcal{P}(E)$. For $P \subseteq E$

$$P \subseteq \nu S.\phi(S) \quad \Leftrightarrow \quad P \subseteq \phi(\nu S.(P \cup \phi(S))).$$

Proof:

"$\Rightarrow$" Assume $P \subseteq \nu S.\phi(S)$. Then

$$P \cup \phi(\nu S.\phi(S)) = P \cup \nu S.\phi(S) = \nu S.\phi(S).$$

Therefore $\nu S.\phi(S)$ is a postfixed point of $S \mapsto P \cup \phi(S)$. As $\nu S.(P \cup \phi(S))$ is the greatest such postfixed point,

$$\nu S.\phi(S) \subseteq \nu S.(P \cup \phi(S)).$$

By monotonicity,

$$\nu S.\phi(S) = \phi(\nu S.\phi(S) \subseteq \phi(\nu S.(P \cup \phi(S)).$$

But $P \subseteq \nu S.\phi(S)$ so $P \subseteq \phi(\nu S(P \cup \phi(S)))$, as required.

"$\Leftarrow$" Assume $P \subseteq \phi(\nu S.(P \cup \phi(S))$. As $\nu S.(P \cup \phi(S))$ is a fixed point of $S \mapsto P \cup \phi(S)$,

$$\nu S.(P \cup \phi(S)) = P \cup \phi(\nu S.(P \cup \phi(S))).$$

Hence, by the assumption

$$\nu S.(P \cup \phi(S)) = \phi(\nu S.(P \cup \phi(S)),$$

i.e. $\nu S.(P \cup \phi(S))$ is a fixed point, and so a postfixed point of ϕ. Therefore

$$\nu S.(P \cup \phi(S)) \subseteq \nu S.\phi(S)$$

as $\nu S.\phi(S)$ is the greatest postfixed point. Clearly $P \subseteq \nu S.(P \cup \phi(S))$ so $P \subseteq \nu S.\phi(S)$, as required. ▮

We are especially concerned with this lemma in the case where P is a singleton set $\{p\}$. In this case the lemma specialises to

$$p \in \nu S.\phi(S) \Leftrightarrow p \in \phi(\nu S.(\{p\} \cup \phi(S))).$$

For us $\nu S.\phi(S)$ will be a property of processes expressed by a recursive assertion and a point, like p, a process. The equivalence says a process

p satisfies a recursively defined property iff the process satisfies a certain kind of unfolding of the recursively defined property. The unfolding is unusual because into the body of the recursion we substitute not just the original recursive definition but instead a recursive definition in which the body is enlarged to contain p. As we shall see, there is a precise sense in which this small modification, $p \in \phi(\nu S.(\{p\} \cup \phi(S)))$, is easier to establish than $p \in \nu S.\phi(S)$, thus providing a method for deciding the truth of recursively defined assertions at a process. The proof systems of [4] and [8] hinge on the same idea though, to my mind, in a rather disguised form. For example, Larsen's proof system for maximum fixed points is essentially based on the backwards-sound proof rules

$$p \in \nu S.(\{p\} \cup \phi(S)) \qquad \frac{p \in \phi(\nu S.(\{p\} \cup \phi(S)))}{p \in \nu S.\phi(S)},$$

though he uses of sequents containing assumptions like $p : S$ instead of process names in the assertions themselves.

2 The modal ν-calculus

We wish to check assertions are true of processes (or states) of a labelled transition system (**Proc**, $\{\xrightarrow{\alpha} \mid \alpha \in \text{labels}\}$). Perhaps there are some basic properties of processes which we wish to keep track of. To cater for this we take a model to have the form

$$(\mathbf{Proc}, \{\xrightarrow{\alpha} \mid \alpha \in \text{labels}\}, V).$$

where V is a function from basic assertions a to the subsets $V(a) \subseteq \mathbf{Proc}$ of processes satisfying them. We assume the model is nonempty and finite, i.e. that **Proc** is a nonempty finite set. With respect to such a model we define the following language of assertions.

$$A ::= a \mid T \mid F \mid \neg A \mid A_0 \wedge A_1 \mid A_0 \vee A_1 \mid \langle\alpha\rangle A \mid X \mid \nu X\{\vec{r}\}A$$

where a is a basic assertion, $\alpha \in$ labels, $X \in \mathbf{Var}$ a set of variables, and $\vec{r} = r_1 \ldots, r_n$ is a finite, possibly empty, list of processes. We shall regard $\{\vec{r}\}$ as a finite, possibly empty, set. A syntactic restriction is placed on the formation of $\nu X\{\vec{r}\}A$ so that A determines a monotonic function according to the semantics, presented shortly: every free occurrence of X in A should be under an even number of negations. A conventional maximum fixed point $\nu X.A$ can be taken to abbreviate $\nu X\{\}A$. Although we have chosen to base the assertion language on maximum fixed points, minimum fixed points can be derived; a minimum fixed point $\mu X.A$ can be defined as $\neg\nu X.\neg A[\neg X/X]$. The assertion $A[\neg X/X]$ is like A but with $\neg X$ substituted for every free occurrence of X. In general $A[B/X]$ is the assertion resulting from substituting B for every free occurrence of X, taking care that no free variables in B become bound in A by renaming bound variables. Fortunately, from now on we shall only deal with substitutions $A[B/X]$ in which B is closed, and so avoid the awkwardness with accidental binding of free variables. Later we shall use $FV(A)$ to mean the set of free variables of A. The set of assertions in general we write as **Assn**, and the set of closed assertions is written as $\mathbf{Assn^c}$.

An environment is a function $\rho : \mathbf{Var} \to \mathcal{P}(\mathbf{Proc})$. For an environment ρ, variable X and subset $S \subseteq \mathbf{Proc}$ we use $\rho[S/X]$ to mean ρ updated to have value S at X i.e.

$$\rho[S/X](Y) = \begin{cases} \rho(Y) & \text{if } Y \not\equiv X, \\ S & \text{if } Y \equiv X \end{cases}$$

The denotation $[\![A]\!]$ of an assertion A is a function from environments to subsets of **Proc** such that:

$$\begin{array}{rcl} [\![a]\!]\rho & = & V(a) \\ [\![T]\!]\rho & = & \mathbf{Proc}, \quad [\![F]\!]\rho = \emptyset, \\ [\![\neg A]\!]\rho & = & \mathbf{Proc} \setminus [\![A]\!]\rho, \\ [\![A_0 \wedge A_1]\!]\rho & = & [\![A_0]\!]\rho \cap [\![A_1]\!]\rho, \\ [\![A_0 \vee A_1]\!]\rho & = & [\![A_0]\!]\rho \cup [\![A_1]\!]\rho, \\ [\![\langle\alpha\rangle A]\!]\rho & = & \{p \in \mathbf{Proc} \mid \exists q \in [\![A]\!]\rho.\ p \xrightarrow{\alpha} q\}, \\ [\![X]\!]\rho & = & \rho(X), \\ [\![\nu X\{\vec{r}\}A]\!]\rho & = & \bigcup\{S \subseteq \mathbf{Proc} \mid S \subseteq \{\vec{r}\} \cup [\![A]\!]\rho[S/X]\}. \end{array}$$

Thus the denotation of $[\![\nu X\{\vec{r}\}A]\!]\rho$ is the maximum fixed point of the function

$$S \mapsto \{\vec{r}\} \cup [\![A]\!]\rho[S/X].$$

In particular $\nu X\{\}A$ denotes the maximum fixed point $\nu S.([\![A]\!]\rho[S/X])$ with respect to an environment ρ. If we were to include basic assertions corresponding to states i.e. for a process p we have an assertion $\hat{p}$ such that $[\![\hat{p}]\!]\rho = \{p\}$ then we could write $\nu X\{r_1, \ldots, r_n\}A$ instead as $\nu X.(\hat{r}_1 \vee \ldots \vee \hat{r}_n \vee A)$. We do not make processes into assertions in this way because the generality is not needed and it would complicate the definition of a well-founded relation on assertions, important for the proof of soundness and completeness of the model checker.

For examples and details of the expressiveness of the ν-calculus we refer the reader to [2], [4], [8] and [10].

We mention a technical lemma to do with substitutions which will be of use later.

Lemma 3 *(substitution lemma)*

For B a closed assertion, X a variable, A an assertion we have

$$[\![A[B/X]]\!]\rho = [\![A]\!]\rho[[\![B]\!]\rho/X].$$

Proof: This is proved to hold for all assertions A by a routine structural induction on A. ▮

We are interested in establishing an algorithm to decide efficiently whether or not a process p satisfies a closed assertion A. In other words, we wish to determine whether $p \in [\![A]\!]\rho$ is true or false, for an arbitrary environment ρ—the environment ρ cannot affect the denotation $[\![A]\!]\rho$ of a closed assertion A.

Definition: Define a *correctness assertion* to be a pair $p \vdash A$ where $p \in \mathbf{Proc}$ and $A \in \mathbf{Assn}^c$. Let ρ be an arbitrary environment. For a correctness assertion $p \vdash A$, define

$$[\![p \vdash A]\!] = \begin{cases} \text{true} & \text{if } p \in [\![A]\!]\rho \\ \text{false} & \text{otherwise.} \end{cases}$$

The algorithm will be presented in the form of reduction rules involving logical combinations of correctness assertions. Ultimately, the reduction works on primitive operations on truth values, the notation for which is defined now.

Notation: Write T for the set of truth values {true, false}. Write $\neg_T$ for the operation of negation on T; thus $\neg_T$ (true) = false and $\neg_T$ (false) = true. Write $\wedge_T$ for the operation of binary conjunction on T; thus $t_0 \wedge_T t_1$ is true if both t_0 and t_1 are true and false otherwise. Write $\vee_T$ for the operation of binary disjunction; thus $t_0 \vee_T t_1$ is true if either t_0 or t_1 is true and false otherwise. It is convenient to have disjunctions of finite sets. Let I be a finite set indexing truth values $\{t_i \mid i \in I\}$. The disjunction $\bigvee_T\{t_i \mid i \in I\}$ is true if t_i is true for some $i \in I$ and false otherwise.

The heart of the reduction of a correctness assertion to a truth value is contained in the next lemma which expresses equalities between the truth values of correctness assertions. (That something is indeed reduced in passing from the left to the right hand side of (vi)(b), so ensuring termination of the reduction, will be demonstrated in the next section.)

Lemma 4 *Let $p \in \mathbf{Proc}$. For closed assertions A, A_0, A_1 and assertion B such that $FV(B) \subseteq \{X\}$ we have:*

$$
\begin{array}{rlrcll}
\text{(i)} & & \llbracket p \vdash T \rrbracket = \mathit{true}, & & \llbracket p \vdash F \rrbracket = \mathit{false} & \\
\text{(ii)} & & \llbracket p \vdash \neg A \rrbracket & = & \neg_T \llbracket p \vdash A \rrbracket & \\
\text{(iii)} & & \llbracket p \vdash A_0 \wedge A_1 \rrbracket & = & \llbracket p \vdash A_0 \rrbracket \wedge_T \llbracket p \vdash A_1 \rrbracket & \\
\text{(iv)} & & \llbracket p \vdash A_0 \vee A_1 \rrbracket & = & \llbracket p \vdash A_0 \rrbracket \vee_T \llbracket p \vdash A_1 \rrbracket & \\
\text{(v)} & & \llbracket p \vdash \langle \alpha \rangle A \rrbracket & = & \bigvee_T \{ \llbracket q \vdash A \rrbracket \mid p \xrightarrow{\alpha} q \} & \\
\text{(vi)} & \text{(a)} & \llbracket p \vdash \nu X \{\vec{r}\} B \rrbracket & = & \mathit{true} \quad \mathit{if}\, p \in \{\vec{r}\} & \\
 & \text{(b)} & \llbracket p \vdash \nu X \{\vec{r}\} B \rrbracket & = & \llbracket p \vdash B[\nu X\{p, \vec{r}\} B / X] \rrbracket & \mathit{if}\, p \notin \{\vec{r}\}
\end{array}
$$

Proof: Parts (i)—(v) are obvious from the definitions. We only show (vi).

(a) Suppose $p \in \{\vec{r}\}$. By definition

$$\llbracket \nu X \{\vec{r}\} B \rrbracket \rho = \bigcup \{ S \subseteq \mathbf{Proc} \mid S \subseteq \{\vec{r}\} \cup \llbracket B \rrbracket \rho[S/X] \}$$

for an arbitrary ρ. Thus $\{\vec{r}\} \subseteq \llbracket \nu X \{\vec{r}\} B \rrbracket \rho$. So $p \in \llbracket \nu X \{\vec{r}\} B \rrbracket \rho$ making
$\llbracket p \vdash \nu X \{\vec{r}\} B \rrbracket = \text{true}$.

(b) Suppose $p \notin \{\vec{r}\}$. Let ρ be an arbitrary environment. Define $\phi(S) = \llbracket B \rrbracket \rho[S/X]$ for $S \subseteq \mathbf{Proc}$. Then $\llbracket \nu X \{\vec{r}\} B \rrbracket \rho = \nu S.\ (\{\vec{r}\} \cup \phi(S))$ by definition. We obtain the following chain of equivalences:

$$
\begin{array}{lll}
\llbracket p \vdash \nu X \{\vec{r}\} B \rrbracket = \text{true} & \text{iff} & p \in \llbracket \nu X \{\vec{r}\} B \rrbracket \rho \\
 & \text{iff} & \{p\} \subseteq \nu S.\ (\{\vec{r}\} \cup \phi(S)) \\
 & \text{iff} & \{p\} \subseteq \{\vec{r}\} \cup \phi(\nu S\ (\{p, \vec{r}\} \cup \phi(S))) \\
 & & \qquad \text{by the reduction lemma 2} \\
 & \text{iff} & p \in \phi(\nu S.(\{p, \vec{r}\} \cup \phi(S))) \text{ as } p \notin \{\vec{r}\} \\
 & \text{iff} & p \in \llbracket B \rrbracket \rho[\nu S.(\{p, \vec{r}\} \cup \llbracket B \rrbracket \rho[S/X])/X] \\
 & & \qquad \text{by recalling the definition of } \phi \\
 & \text{iff} & p \in \llbracket B \rrbracket \rho[\llbracket \nu X \{p, \vec{r}\} B \rrbracket \rho / X] \\
 & & \qquad \text{from the definition of } \llbracket \nu X \{\vec{r}\} B \rrbracket \\
 & \text{iff} & p \in \llbracket B[\nu X \{p, \vec{r}\} B / X] \rrbracket \rho \\
 & & \qquad \text{by the substitution lemma 3} \\
 & & \text{iff } \llbracket p \vdash B[\nu X \{p, \vec{r}\} B / X] \rrbracket = \text{true}.
\end{array}
$$

Thus $[\![p \vdash \nu X\{\vec{r}\}B]\!] = \text{true}$ iff $[\![p \vdash B[\nu X\{p, \vec{r}\}B/X]]\!] = \text{true}$, if $p \notin \{\vec{r}\}$. As $[\![p \vdash A]\!]$ has value true or false for any closed assertion A, this implies (b). ▮

The model checker

The last lemma suggests a reduction strategy for settling whether or not a process p satisfies a closed assertion A. With respect to the model $(\mathbf{Proc}, \{\xrightarrow{\alpha} \mid \alpha \in \text{Labels}\}, V)$ we mimic the equations of the lemma by the following reduction rules:

$$\begin{array}{rcll}
(p \vdash a) & \to & \text{true} & \text{if } p \in V(a) \\
(p \vdash a) & \to & \text{false} & \text{if } p \notin V(a) \\
(p \vdash T) & \to & \text{true} & \\
(p \vdash F) & \to & \text{false} & \\
(p \vdash \neg B) & \to & \neg(p \vdash B) & \\
(p \vdash A_0 \wedge A_1) & \to & (p \vdash A_0) \wedge (p \vdash A_1) & \\
(p \vdash A_0 \vee A_1) & \to & (p \vdash A_0) \vee (p \vdash A_1) & \\
(p \vdash \langle\alpha\rangle B) & \to & \bigvee\{q \vdash B \mid p \xrightarrow{\alpha} q\} & \\
(p \vdash \nu X\{\vec{r}\}B) & \to & \text{true} & \text{if } p \in \{\vec{r}\} \\
(p \vdash \nu X\{\vec{r}\}B) & \to & (p \vdash B[\nu X\{p, \vec{r}\}B/X]) & \text{if } p \notin \{\vec{r}\}
\end{array}$$

A full set of reduction rules will also include rules specifying the evaluation of the boolean expressions built out of correctness assertions using $\neg, \wedge, \vee$ and $\bigvee$. To cover the range of different methods of evaluation of such boolean expressions we merely stipulate that the rules have the following properties:

For negations:

$$\forall t \in T.\ (b \to^* t \Leftrightarrow \neg b \to^* \neg_T t)$$

For conjunctions:
If $b_0 \to^* t_0$ and $b_1 \to^* t_1$ and $t_0, t_1 \in T$ then

$$\forall t \in T.\ (b_0 \wedge b_1 \to^* t \Leftrightarrow t_0 \wedge_T t_1 = t)$$

For disjunctions:
If $b_0 \to^* t_0$ and $b_1 \to^* t_1$ and $t_0, t_1 \in T$ then

$$\forall t \in T.\ (b_0 \vee b_1 \to^* t \Leftrightarrow t_0 \vee_T t_1 = t)$$

For disjunctions of sets:
If $\forall i \in I.\ (b_i \longrightarrow^* t_i$ and $t_i \in T)$ then

$$\forall t \in T.\ \bigvee\{b_i \mid i \in I\} \longrightarrow^* t \Leftrightarrow \bigvee_T\{t_i \mid i \in I\} = t$$

Certainly, any sensible rules for the evaluation of boolean expressions will have the properties above, whether the evaluation proceeds in a left-to-right, right-to-left or parallel fashion. Under these assumptions the reduction rules are sound and complete in the sense of the theorem below.

Theorem 5 *Let $p \in$ **Proc** and A be a closed assertion. For any truth value $t \in T$,*

$$(p \vdash A) \longrightarrow^* t \ \text{ iff } [\![p \vdash A]\!] = t.$$

Proof: The proof proceeds by well-founded induction on assertions with the relation

$$\begin{aligned} A' \prec A \text{ iff } & A' \text{ is a proper subassertion of } A \\ & \text{or } A, A' \text{ have the form} \\ & A \equiv \nu X\{\vec{r}\}B \text{ and } A' \equiv \nu X\{p, \vec{r}\}B \text{ with } p \notin \{\vec{r}\} \end{aligned}$$

As **Proc** is a finite set, the relation $\prec$ is well-founded.

We are interested in showing the property

$$Q(A) \Leftrightarrow_{def} \forall p \in \textbf{Proc}\ \forall t \in T.\ [(p \vdash A) \longrightarrow^* t \Leftrightarrow [\![p \vdash A]\!] = t]$$

holds for all closed assertions A. The proof however requires us to extend the property Q to assertions A with free variables $FV(A)$, which we do in the following way:
For $A \in$ **Assn** ,

$$\begin{aligned} Q^+(A) \Leftrightarrow_{def} \forall \theta : FV(A) \rightarrow \textbf{Assn}^{\,c}. \\ [(\forall X \in FV(A).\ Q(\theta(X))) \Rightarrow Q(A[\theta])]. \end{aligned}$$

Notice that when A is closed $Q^+(A)$ is logically equivalent to $Q(A)$.

We show $Q^+(A)$ holds for all assertions A by well-founded induction on $\prec$. To this end let A be an assertion such that $Q^+(A')$ for all assertions $A' \prec A$. We are required to show it follows that $Q^+(A)$. So we let $\theta : FV(A) \rightarrow \textbf{Assn}^{\,c}$ with $\forall X \in FV(A).\ Q(\theta(X))$ and show $Q(A[\theta])$ for all the possible forms of A.

$A \equiv a$: In the case where $A \equiv a$, a basic assertion, we have $A[\theta] \equiv a$ and

$$(p \vdash a) \longrightarrow^* t \Leftrightarrow \llbracket p \vdash a \rrbracket = t$$

for all $p \in \mathbf{Proc}, t \in T$. Hence $Q(A[\theta])$.

$A \equiv T$: In this case, $A[\theta] \equiv T$, and

$$\begin{aligned}(p \vdash T) \longrightarrow^* t \ &\Leftrightarrow \ \mathrm{true} \equiv t \\ &\Leftrightarrow \ \llbracket p \vdash T \rrbracket = t\end{aligned}$$

for all $p \in \mathbf{Proc}, t \in T$. Hence $Q(A[\theta])$ in this case.

$A \equiv F$: In this case, $A[\theta] \equiv F$, and

$$\begin{aligned}(p \vdash F) \longrightarrow^* t \ &\Leftrightarrow \ \mathrm{false} \equiv t \\ &\Leftrightarrow \ \llbracket p \vdash F \rrbracket = t\end{aligned}$$

for all $p \in \mathbf{Proc}, t \in T$. Hence $Q(A[\theta])$.

$A \equiv \neg B$: In this case $A[\theta] \equiv \neg(B[\theta])$. As $B \prec A$ we have $Q^+(B)$, so $Q(B)$. Letting $p \in \mathbf{Proc}, t \in T$ we have:

$$\begin{aligned}(p \vdash \neg(B[\theta]) \longrightarrow^* t \ &\Leftrightarrow \ \neg(p \vdash B[\theta]) \longrightarrow^* t \\ &\Leftrightarrow \ (p \vdash B[\theta]) \longrightarrow^* \neg_T t && \text{by the property assumed of evaluation for negations} \\ &\Leftrightarrow \ \llbracket p \vdash B[\theta] \rrbracket = \neg_T t && \text{as } Q(B[\theta]) \\ &\Leftrightarrow \ \llbracket p \vdash \neg(B[\theta]) \rrbracket = t\end{aligned}$$

Hence $Q(A[\theta])$ in this case.

$A \equiv A_0 \wedge A_1$: In this case $A[\theta] \equiv A_0[\theta] \wedge A_1]\theta]$. Let $p \in \mathbf{Proc}$. Let $\llbracket p \vdash A_0[\theta] \rrbracket = t_0$ and $\llbracket p \vdash A_1[\theta] \rrbracket = t_1$. As $A_0 \prec A$ and $A_1 \prec A$ we have $Q^+(A_0)$ and $Q^+(A_1)$. Thus $Q(A_0[\theta])$ and $Q(A_1[\theta])$, so $(p \vdash A_0[\theta]) \longrightarrow^* t_0$ and $(p \vdash A_1[\theta]) \longrightarrow^* t_1$. Now, for $t \in T$,

$$\begin{aligned}(p \vdash A_0[\theta] \wedge A_1[\theta])) \longrightarrow^* t \ &\Leftrightarrow \ (p \vdash A_0[\theta]) \wedge (p \vdash A_1[\theta])) \longrightarrow^* t \\ &\Leftrightarrow \ t_0 \wedge_T t_1 = t \\ & \ \ \text{by the property assumed for the evaluation of conjunctions} \\ &\Leftrightarrow \ \llbracket p \vdash A_0[\theta] \rrbracket \wedge_T \llbracket p \vdash A_1[\theta] \rrbracket = t \\ &\Leftrightarrow \ \llbracket p \vdash A_0[\theta] \wedge A_1[\theta] \rrbracket = t\end{aligned}$$

Hence $Q(A[\theta])$ in this case.

$A \equiv A_0 \vee A_1$: In this case $A[\theta] \equiv A_0[\theta] \vee A_1]\theta]$. As $A_0 \prec A$ and $A_1 \prec A$ we have $Q(A_0[\theta])$ and $Q(A_1[\theta])$. Let $p \in \mathbf{Proc}$. Letting $[\![p \vdash A_0[\theta]]\!] = t_0$ and $[\![p \vdash A_1[\theta]]\!] = t_1$, we obtain $(p \vdash A_0[\theta]) \rightarrow^* t_0$ and $(p \vdash A_1[\theta]) \rightarrow^* t_1$. Now, for $t \in T$,

$$\begin{aligned}(p \vdash A_0[\theta] \vee A_1[\theta])) \rightarrow^* t \quad &\Leftrightarrow \quad (p \vdash A_0[\theta]) \vee (p \vdash A_1[\theta])) \rightarrow^* t\\ &\Leftrightarrow \quad t_0 \vee_T t_1 = t\\ &\qquad \text{by the property assumed for the evalua-}\\ &\qquad \text{tion of disjunctions}\\ &\Leftrightarrow \quad [\![p \vdash A_0[\theta]]\!] \vee_T [\![p \vdash A_1[\theta]]\!] = t\\ &\Leftrightarrow \quad [\![p \vdash A_0[\theta] \vee A_1[\theta]]\!] = t\end{aligned}$$

Hence $Q(A[\theta])$ in this case.

$A \equiv \langle\alpha\rangle B$: In this case $A[\theta] = \langle\alpha\rangle(B[\theta])$. Let $p \in \mathbf{Proc}$. For all q such that $p \xrightarrow{\alpha} q$ let $[\![q \vdash B[\theta]]\!] = t_q$. As $B \prec A$ we have $Q(B[\theta])$ and thus $(q \vdash B[\theta]) \rightarrow^* t_q$ for all q such that $p \xrightarrow{\alpha} q$. Now, for $t \in T$,

$$\begin{aligned}(p \vdash \langle\alpha\rangle(B[\theta])) \rightarrow^* t \quad &\Leftrightarrow \quad \bigvee\{(q \vdash B[\theta]) \mid p \xrightarrow{\alpha} q\} \rightarrow^* t\\ &\Leftrightarrow \quad \bigvee_T\{t_q \mid p \xrightarrow{\alpha} q\} = t\\ &\qquad \text{by the property assumed for}\\ &\qquad \text{the evaluation of disjunctions}\\ &\Leftrightarrow \quad \bigvee_T\{[\![q \vdash B[\theta]]\!] \mid p \xrightarrow{\alpha} q\} = t\\ &\Leftrightarrow \quad [\![p \vdash \langle\alpha\rangle(B[\theta])]\!] = t\end{aligned}$$

Hence $Q(A[\theta])$ in this case.

$A \equiv X$: In this case, when A is a variable the $Q(A[\theta])$ holds trivially.

$A \equiv \nu X\{\vec{r}\}B$: In this case $A[\theta] \equiv \nu X\{\vec{r}\}(B[\theta])$. Let $p \in \mathbf{Proc}$. Either $p \in \{\vec{r}\}$ or not.

If $p \in \{\vec{r}\}$ then

$$\begin{aligned}(p \vdash \nu X\{\vec{r}\}(B[\theta])) \rightarrow^* t \quad &\Leftrightarrow \quad \text{true} = t\\ &\Leftrightarrow \quad [\![p \vdash \nu X\{\vec{r}\}(B[\theta])]\!] = t\end{aligned}$$

for any $t \in T$.

Otherwise $p \notin \{\vec{r}\}$. Then $\nu X\{p,\vec{r}\}B \prec A$, so $Q(\nu X\{p,\vec{r}\}(B[\theta]))$. Define
$\theta' : FV(B) \rightarrow$ **Assn** ^{c} by

$$\theta'(Y) = \begin{cases} \theta(Y) & \text{if } Y \not\equiv X \\ \nu X\{p,\vec{r}\}(B[\theta]) & \text{if } Y \equiv X \end{cases}$$

for $Y \in FV(B)$. Certainly $\forall Y \in FV(B).\ Q(\theta'(Y))$. As $B \prec A$ we have $Q^+(B)$. Hence $Q(B[\theta'])$. But $B[\theta'] \equiv (B[\theta])[\nu X\{p,\vec{r}\}(B[\theta])/X]$. Thus from the reduction rules,

$$\begin{aligned}(p \vdash \nu X\{\vec{r}\}(B[\theta])) \rightarrow^* t &\Leftrightarrow (p \vdash (B[\theta])[\nu X\{p,\vec{r}\}(B[\theta])/X]) \rightarrow^* t \\ &\Leftrightarrow (p \vdash B[\theta']) \rightarrow^* t \\ &\Leftrightarrow [\![p \vdash B[\theta']]\!] = t \quad \text{as } Q(B[\theta']) \\ &\Leftrightarrow [\![p \vdash (B[\theta])[\nu X\{p,\vec{r}\}(B[\theta])/X]]\!] = t \\ &\Leftrightarrow [\![p \vdash \nu X\{\vec{r}\}(B[\theta])]\!] = t \quad \text{by lemma 4.}\end{aligned}$$

Hence, whether $p \in \{\vec{r}\}$ or not, $Q(A[\theta])$ in this case.

In every case determined by the form of A we have shown $Q(A[\theta])$. It follows that $\forall A \in$ **Assn**. $(\forall B \prec A \quad Q^+(B)) \Rightarrow Q^+(A)$. By well-founded induction we conclude $Q^+(A)$ for all assertions A, and in particular $Q(A)$ for all closed assertions A. ▌

As a corollary we obtain the termination of the algorithm on all correctness assertions. In particular the evaluation of boolean expressions in standard ML or Miranda have the properties we require for the theorem to hold, and in these languages it is straightforward to write a function which evaluates according to the reduction rules, and thus obtain a working model checker. Of course, it is not tuned for efficiency and, beyond some keeping-track of the states visited, makes no optimisations. Some discussion of suitable optimisations and the comparison with other algorithms can be found in [1]. In particular, [1] shows that the approach of "local model checking" does not directly improve on the asymptotic complexity of the algorithm in [2]. However, this is in the worst cases; the algorithm of [2] finds *all* the states that satisfy an assertion, which is often unnecessary in determining whether or not a particular state does and can be avoided with the approach of "local model checking".

Acknowledgements

This note can be viewed as a recasting of some of the work for the Edinburgh-Sussex Concurrency Workbench done by Kim Larsen, Colin Stirling and David Walker, albeit with improvements and simplifications. Around the time that this note was being prepared Rance Cleaveland used very similar ideas to give an another tableau method [1] for model checking the ν-calculus but one which was more easily implementable than that of Stirling and Walker.

References

[1] Cleaveland, R., Tableau-based model checking in the propositional μ-calculus. Report of the Computer Science Department, University of Sussex, 1989. To appear in Acta Informatica.

[2] Emerson, A. and Lei, C., Efficient model checking in fragments of the propositional mu-calculus. Proc.LICS, 1986.

[3] Cleaveland, R., Parrow, J. and Steffen, B., The Concurrency Workbench. To appear.

[4] Larsen, K.G., Proof systems for Hennessy-Milner logic. Proc. CAAP, 1988.

[5] Kozen, D., Results on the propositional mu-calculus. Theor.Comp.Sci.27, 333-354, 1983.

[6] Milner, A.J.R.G., Communication and concurrency. Prentice Hall, 1989.

[7] Pratt, V., A decidable μ-calculus. Proc.22nd.FOCS, 421-27, 1981.

[8] Stirling, C. and Walker D., Local model checking the modal mu-calculus. Proc.CAAP, 1989.

[9] Tarski, A., A lattice-theoretical fixpoint theorem and its applications. Pacific Journal of Mathematics, 5, 1955.

[10] Walker, D., Automated analysis of mutual exclusion algorithms using CCS. Submitted for publication, 1988.