Verifying Temporal Properties of Processes

Julian Bradfield, Colin Stirling
Department of Computer Science
University of Edinburgh
The King's Buildings
EDINBURGH
U.K. EH9 3JZ
email: jcb@lfcs.ed.ac.uk, cps@lfcs.ed.ac.uk

1 Motivation

Many interesting concurrent systems have infinite state spaces: examples include concurrent while programs; Petri Nets; CCS (or CSP) processes with value passing. All of these examples can be interpreted operationally as infinite labelled transition systems, structures of the form $(\mathcal{P}, \{ \xrightarrow{a} : a \in L \})$ where $\mathcal{P}$ is a set of points (states, markings, processes) and $\xrightarrow{a}$ the appropriate binary transition relation on $\mathcal{P}$ for each label a (action, set of events) belonging to the family L.

A very rich temporal logic for expressing properties of such transition systems is a slight extension of the modal mu-calculus [9, 6] where the modalities are indexed by families of labels instead of individual labels. The question we address in this paper is: can model checking techniques, as introduced in [3] be extended from finite to infinite state spaces? (Pragmatically, this means moving from automated to computer-aided verification techniques.) We provide an affirmative answer by presenting a *sound and complete* tableau system for proving temporal properties of states (processes or markings) in *arbitrary* infinite transition system models. The tableau system extends local model checking techniques as presented in [4, 7, 11, 14]. The delicate aspect is showing that a point (or set of points) has, or lacks, a least fixed point property (a *liveness* property). The tableau proof system is data independent and therefore generalizes standard methods commonly used in program logics (such as Hoare logics). The verification technique is illustrated on examples drawn from CCS [8]. See [1] for the application of the method to Petri Nets and [12] for its application to concurrent while programs.

Section 2 provides examples of CCS processes and their properties. In section 3 the syntax and semantics of the (slightly extended) modal mu-calculus are described. The tableau proof system is presented in section 4, and finally we briefly examine applications in section 5.

2 Processes

Processes of CCS determine labelled transition systems under their behavioural (transitional) semantics. In this case $\xrightarrow{a}$ is the usual CCS derivation relation with $E \xrightarrow{a} F$ representing that process E may become F by performing a. When L is the set of all actions, and $K \subseteq L$, we write $\xrightarrow{K}$ for $\bigcup_{a \in K} \xrightarrow{a}$. Moreover we write $\xRightarrow{a}$ for the observable derivation, defined as usual by

$$\begin{array}{lll} E \xRightarrow{\epsilon} F & \text{if} & E = E_0 \xrightarrow{\tau} E_1 \xrightarrow{\tau} \cdots \xrightarrow{\tau} E_n = F \quad (n \geq 0) \\ E \xRightarrow{a} F & \text{if} & E \xRightarrow{\epsilon} E' \quad \text{and} \quad E' \xrightarrow{a} F' \quad \text{and} \quad F' \xRightarrow{\epsilon} F \end{array}$$

We now present four example processes, and some liveness and safety properties that may be ascribed to them.

First, we have a simple finite state system, representing a road crossing a railway; unlike most crossings, this one keeps the barriers down except when a car actually approaches and tries to cross.

$$\begin{array}{lcl} Rail & \stackrel{def}{=} & train.green.tcross.\overline{red}.Rail \\ Road & \stackrel{def}{=} & car.up.ccross.\overline{down}.Road \\ Signal & \stackrel{def}{=} & \overline{green}.red.Signal + \overline{up}.down.Signal \\ Crossing & \stackrel{def}{=} & (Road \mid Rail \mid Signal)\backslash\{green, red, up, down\} \end{array}$$

Here, $train$ and car represent the approach of a train and car respectively, $green$ is the receipt of a green signal by the train, $tcross$ is the train crossing and $\overline{red}$ automatically sets the lights red, up is the gates opening for the car, and $ccross$ is the car crossing and $\overline{down}$ closes the gates. A crucial safety property of this system is that it is never possible for a train and a car both to be able to cross. A very desirable liveness property (which the crossing fails to have) is that whenever a car or train approaches, it eventually crosses. However, it does have this property if we assume that the signal is fair.

For a simple infinite state system, consider a process $Ticker$ whose observable behaviour is to perform a finite number of $tick$s and then stop.

$$\begin{array}{lcll} Up_i & \stackrel{def}{=} & \tau.Up_{i+1} + Down_i & (i \geq 0) \\ Down_i & \stackrel{def}{=} & tick.Down_{i-1} & (i \geq 1) \\ Down_0 & \stackrel{def}{=} & 0 & \\ Ticker & \stackrel{def}{=} & Up_0 & \end{array}$$

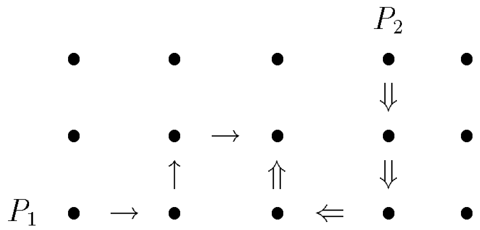
This process has the feature that it may diverge, that is, perform τs indefinitely. So the expression of "*Ticker* performs a finite number of ticks and then stops" must account for this; and we should also like to express the stronger property that "*Ticker* may diverge initially, but once it starts ticking, it does nothing else until it stops".

A more complex example (employing value passing) is a slot machine SM_n, with three components—IO, which handles taking and paying out money, B_n a bank holding n pounds, and D, the wheel-spinning decision component.

$$
\begin{array}{lcl}
IO & \stackrel{def}{=} & slot.\overline{bank}.(lost.\overline{loss}.IO + release(y).\overline{win}(y).IO) \\
B_n & \stackrel{def}{=} & bank.\overline{max}(n+1).left(y).B_y \\
D & \stackrel{def}{=} & max(z).\big(\overline{lost}.\overline{left}(z).D + \sum_{1 \le y \le z} \overline{release}(y).\overline{left}(z-y).D\big) \\
SM_n & \stackrel{def}{=} & (IO \mid B_n \mid D)\backslash K
\end{array}
$$

where $K = \{\, bank, max(v), left(v), release(v) : v \in \mathbb{N} \,\}$. Again, we have a safety property, that the bank never has a negative amount of money, and a weak liveness property, that it is possible to win a million pounds infinitely often.

Finally, we have a complex example illustrating some of the issues raised by networks of similar processes. Consider a set of n processes placed on a grid of relay stations which allow them to communicate. Suppose that each process wishes to establish a dedicated and private communication link with each other process: to do this, each pair of processes requires a chain of dedicated relay stations between them. Since each relay can only communicate with its four immediate neighbours, some method is needed to achieve this. We shall consider one of the simplest: each process takes control of the $n-1$ (say) stations initially assigned to it, and instructs them to take control of a random neighour; communication is established when two such chains owned by different processes meet head on, as in the picture:



We shall suppose that each process is assigned a line of $n-1$ adjacent stations.

A crude model of such a system is given by letting $\{C_{ij}\}$ be an array of cells, each defined by

$$
\begin{array}{lcl}
C_{ij} & \stackrel{def}{=} & alloc_{ij}(k).(\overline{alloc}_{i\,j+1}(k).0 + \overline{alloc}_{i+1\,j}(k).0 + \overline{alloc}_{i\,j-1}(k).0 + \overline{alloc}_{i-1\,j}(k).0 \\
 & & +alloc_{ij}(k').if\ k = k'\ then\ 0\ else\ \overline{line}(min(k,k'), max(k.k')))
\end{array}
$$

That is, a cell waits to be allocated to some process k, and then tries to allocate its neighbours to k, and also listens to see if a different process k' wants this cell. If it allocates a neighbour, the cell plays no further part; if it is contacted by process k', it signals the successful establishment of a line from k to k'.

Suppose that process P_k is initially assigned cells $C_{i_k j_k+1}, \ldots, C_{i_k j_k+n-1}$: P_k is then just

$$P_k \stackrel{def}{=} \big|_{1 \leq l < n} \overline{alloc}_{i_k\, j_k+l}(k).0$$

The system is then the parallel composition of all cells and processes, restricted on the *alloc* actions. One would like to be able to say that every pair of processes connects, but that is clearly too much to expect of our simple random algorithm, so we shall instead consider the weak liveness property that it is possible for all pairs to eventually connect, and we shall show that this holds only if there are fewer than five processes.

3 The modal mu-calculus

The temporal logic used is (a slight extension of) the modal mu-calculus. The syntax for formulae Φ is as follows:

$$\Phi \ ::= \ Z \mid \neg\Phi \mid \Phi_1 \wedge \Phi_2 \mid [K]\Phi \mid \nu Z.\Phi$$

where Z ranges over a denumerable family of propositional variables, and K over *subsets* of a label set L. A restriction on $\nu Z.\Phi$ is that each free occurrence of Z in Φ should lie within the scope of an even number of negations. Derived operators are defined in the familiar way: $tt \stackrel{def}{=} \nu Z.Z$; $ff \stackrel{def}{=} \neg tt$; $\Phi \vee \Psi \stackrel{def}{=} \neg(\neg\Phi \wedge \neg\Psi)$; $\langle K\rangle\Phi \stackrel{def}{=} \neg[K]\neg\Phi$; and $\mu Z.\Phi \stackrel{def}{=} \neg\nu Z.\neg\Phi[Z := \neg Z]$ where $\neg\Phi[Z := \neg Z]$ is the result of substituting $\neg Z$ for each free occurrence of Z. Further useful abbreviations (which also apply to the $\langle K\rangle$ modalities) are:

$$\begin{array}{lcl} [-K]\Phi & \stackrel{def}{=} & [L-K]\Phi \\ [a_1,\ldots,a_n]\Phi & \stackrel{def}{=} & [\{a_1,\ldots,a_n\}]\Phi \\ [-]\Phi & \stackrel{def}{=} & [L]\Phi \end{array}$$

Formulae of this logic are interpreted on labelled transition systems $\mathcal{T} = (\mathcal{P}, \{\stackrel{a}{\longrightarrow}: a \in L\})$. A model is a pair $(\mathcal{T}, \mathcal{V})$ where $\mathcal{T}$ is a transition system and $\mathcal{V}$ a valuation assigning sets of processes to propositional variables: $\mathcal{V}(Z) \subseteq \mathcal{P}$. We assume the customary updating notation: $\mathcal{V}[\mathcal{E}/Z]$ is the valuation $\mathcal{V}'$ which agrees with $\mathcal{V}$ except that $\mathcal{V}'(Z) = \mathcal{E}$. Finally the set of processes of $\mathcal{T}$ having the property Φ in the model $(\mathcal{T}, \mathcal{V})$ is inductively defined as $\|\Phi\|^{\mathcal{T}}_{\mathcal{V}}$ (where for ease of notation we drop the index $\mathcal{T}$ which is assumed to be fixed):

$$\begin{array}{lcl} \|Z\|_{\mathcal{V}} & = & \mathcal{V}(Z) \qquad (Z \in Var) \\ \|\neg\Phi\|_{\mathcal{V}} & = & \mathcal{P} - \|\Phi\|_{\mathcal{V}} \\ \|\Phi_1 \wedge \Phi_2\|_{\mathcal{V}} & = & \|\Phi_1\|_{\mathcal{V}} \cap \|\Phi_2\|_{\mathcal{V}} \\ \|[K]\Phi\|_{\mathcal{V}} & = & \{\, P \in \mathcal{P} : \forall P'.\, P \stackrel{K}{\longrightarrow} P' \;\Rightarrow\; P' \in \|\Phi\|_{\mathcal{V}} \,\} \\ \|\nu Z.\Phi\|_{\mathcal{V}} & = & \bigcup\{\, \mathcal{E} \subseteq \mathcal{P} : \|\Phi\|_{\mathcal{V}[\mathcal{E}/Z]} \supseteq \mathcal{E} \,\} \end{array}$$

The expected clause for the derived operator $\mu Z.\Phi$ is:

$$\|\mu Z.\Phi\|_{\mathcal{V}} \;=\; \bigcap\{\, \mathcal{E} \subseteq \mathcal{P} : \|\Phi\|_{\mathcal{V}[\mathcal{E}/Z]} \subseteq \mathcal{E} \,\}$$

CCS processes are interpreted operationally as transition systems. The modal mu-calculus is a natural extension of Hennessy-Milner logic [5], offering a succinct temporal logic for processes. Moreover its set of closed formulas (those without free variables) also characterizes bisimulation equivalence [10]. The examples below illustrate the benefit of extending the modal mu-calculus to encompass modalities with sets of labels. In the general case it allows one to express temporal properties of processes engaged in value passing in a very simple and succinct fashion. Moreover, it furnishes a direct relationship with standard temporal logics such as CTL and LT, where labels are ignored, via the modalities $[-]$ and $\langle - \rangle$.

Assume any mu-calculus model for the *Crossing*. The safety property that it is never possible for both a train and a car both to be able to cross is expressed by the formula

$$\nu Z.([tcross]ff \vee [ccross]ff) \wedge [-]Z$$

More generally, $\nu Z.\Phi \wedge [-]Z$ expresses that Φ is invariantly true. The desirable liveness property that whenever a car approaches then eventually it crosses only holds if we assume that the signal is fair. So let Q be a propositional variable which is true when the crossing is in any state where *Road* has the form $up.ccross.\overline{down}.Road$ and let R hold when it is in a state where *Rail* has the form $green.tcross.\overline{red}.Rail$. ($Q$ and R can be thought of as probes in the sense of [13].) The liveness property is now: for any run if $\neg Q$ is true infinitely often and $\neg R$ is also true infinitely often then whenever a car approaches eventually it crosses. This is expressed by the following formula Φ employing embedded fixed point subformulae:

$$\begin{aligned} \Phi &= \nu Y.[car]\Phi_1 \wedge [-]Y \\ \Phi_1 &= \mu X.\nu Y_1.(Q \vee [-ccross](\nu Y_2.(R \vee X) \wedge [-ccross]Y_2)) \wedge [-ccross]Y_1 \end{aligned}$$

In general for arbitrary models, the property that eventually action *a happens* (as opposed to *may* happen) is given by the formula $\mu Z.\langle - \rangle tt \wedge [-a]Z$.

Another example CCS process from the previous section is *Ticker*. Its *observable* behaviour is to perform a finite number of ticks and then to stop. This description relates to its behaviour with respect to the $\stackrel{a}{\Longrightarrow}$ transitions. So it is useful to introduce the abbreviation

$$[\![\,]\!]\Phi \stackrel{def}{=} \nu Z.\Phi \wedge [\tau]Z$$

(where Z does not occur free in Φ). In a model process E has the property $[\![\,]\!]\Phi$ just in case every process F such that $E \stackrel{\epsilon}{\Longrightarrow} F$ has the property Φ. Then when $\tau \notin K$ the modality $[\![K]\!]$ is defined as $[\![\,]\!][K][\![\,]\!]$: consequently, $[\![K]\!]$ relates to $\stackrel{a}{\Longrightarrow}$ as $[K]$ relates to $\stackrel{a}{\longrightarrow}$. The property above of its observable behaviour is expressed as:

$$\mu Z.[\![tick]\!]Z$$

The ticker may also diverge initially, that is it may perform τs indefinitely. Divergence is given by the formula $\nu Z.\langle\tau\rangle Z$.

In the case of the slot machines SM_n, assume a model whose valuation assigns to the propositional variables Q_i the set $\{ SM_j : j < i \}$ for each $i \geq 0$. Now an important safety property is that SM_n never has a negative amount of money, so never pays out more than it contains:

$$\nu Z.\neg Q_0 \wedge [-]Z$$

A useful weak liveness property is that it may pay out a million (pounds) infinitely often, expressible as

$$\nu Y.\mu Z.(\langle\overline{win}(10^6)\rangle Y) \vee \langle -\rangle Z$$

The weak liveness property for the process network is

$$\mu Z.\Psi \vee \langle -\rangle Z$$

where $\Psi \stackrel{def}{=} \bigwedge_{1\leq j<i\leq n}\langle\overline{line}(j,i)\rangle tt$. This formula is stronger than necessary: all we need is that all $\overline{line}(k,k')$ actions should happen eventually, in any order; although this can be expressed in the mu-calculus, the resulting formula is large; see section 5 for further discussion.

These examples show that we are interested in whether a particular process (state or point) or set of processes $\mathcal{E}$ has a temporal property Φ. But the meaning of Φ is defined in terms of *all* processes having it as a property. Clearly, in general *computing* the set $\|\Phi\|^{\mathcal{T}}_{\mathcal{V}}$ is not only unnecessarily cumbersome (if one is just interested in whether $\mathcal{E} \subseteq \|\Phi\|^{\mathcal{T}}_{\mathcal{V}}$), but also is not really feasible either for large formulae involving numerous embeddings of the fixed point operators, or for very large (let alone infinite) transition systems. Discovering fixed point sets in general is not easy, and is therefore liable to lead to errors. Instead we would like simpler (and consequently safer) methods for verifying that processes have, or fail to have, temporal properties.

Here a temporal property checker is developed as a tableau system in the style of [11] but extended to cope with arbitrary *infinite* transition system models. The rules turn out to be straightforward, but the successful termination condition for least fixed point subformulae is subtle.

4 The tableau system

Assume a fixed mu-calculus model $(\mathcal{T},\mathcal{V})$ where $\mathcal{T}$ may have infinite size. We wish to show that some process, or more generally some set of processes $\mathcal{E}$ has the property Φ, that $\mathcal{E} \subseteq \|\Phi\|^{\mathcal{T}}_{\mathcal{V}}$. We present a tableau system for proving this, built on sequents of the form $\mathcal{E} \vdash_{\Delta} \Phi$ where Δ is a *definition list* which keeps track of unrolling fix-point formulae. A definition is given as $U = \Psi$ where U is a propositional constant and Ψ a fixed point formula. A finite sequence Δ of definitions $(U_1 = \Psi_1, \ldots, U_n = \Psi_n)$ has the two properties that each U_i is distinct, and secondly that each Ψ_i can only mention propositional constants belonging to the set $\{U_1, \ldots, U_{i-1}\}$.[1] Lists of definitions can be extended: if U is not declared in Δ and Ψ only mentions constants declared in Δ, then $\Delta \cdot U = \Psi$ is the definition list that results from appending $U = \Psi$ to Δ. When Δ is a definition list and U is declared to be Ψ in Δ, then $\Delta(U)$ is Ψ.

The rules of the tableau below are inverse natural deduction type rules. The premise sequent is the goal to be achieved (that $\mathcal{E} \subseteq \|\Phi_\Delta\|^{\mathcal{T}}_{\mathcal{V}}$) while the consequents are the subgoals. The rules are presented only for formulae in positive form (where all negations are moved inwards by using the dual operators $\vee$, $\langle K \rangle$ and $\mu Y.$). We assume that σ ranges over ν or μ. In the Diamond rule we assume that f is a function from the set $\mathcal{E}$ to the set $f(\mathcal{E})$ such that for all $E \in \mathcal{E}$, $E \xrightarrow{K} f(E)$:

And	$\dfrac{\mathcal{E} \vdash_\Delta \Phi_1 \wedge \Phi_2}{\mathcal{E} \vdash_\Delta \Phi_1 \qquad \mathcal{E} \vdash_\Delta \Phi_2}$	
Or	$\dfrac{\mathcal{E} \vdash_\Delta \Phi_1 \vee \Phi_2}{\mathcal{E}_1 \vdash_\Delta \Phi_1 \qquad \mathcal{E}_2 \vdash_\Delta \Phi_2}$	$\mathcal{E} = \mathcal{E}_1 \cup \mathcal{E}_2$
Box	$\dfrac{\mathcal{E} \vdash_\Delta [K]\Phi}{\mathcal{E}' \vdash_\Delta \Phi}$	$\mathcal{E}' = \{\, P' : \exists P \in \mathcal{E} \,.\, P \xrightarrow{K} P' \,\}$
Diamond	$\dfrac{\mathcal{E} \vdash_\Delta \langle K \rangle \Phi}{f(\mathcal{E}) \vdash_\Delta \Phi}$	
Constant Introduction	$\dfrac{\mathcal{E} \vdash_\Delta \sigma Z.\Phi}{\mathcal{E} \vdash_{\Delta'} U}$	$\Delta' = \Delta \cdot (U = \sigma Z.\Phi)$
Constant Unfolding	$\dfrac{\mathcal{E} \vdash_\Delta U}{\mathcal{E} \vdash_\Delta \Phi[Z := U]}$	$\Delta(U) = \sigma Z.\Phi$
Thinning	$\dfrac{\mathcal{E} \vdash_\Delta \Phi}{\mathcal{E}' \vdash_\Delta \Phi}$	$\mathcal{E}' \supseteq \mathcal{E}$

To test if every process in $\mathcal{E}$ has the property Φ (relative to Δ) one has to achieve the goal $\mathcal{E} \vdash_\Delta \Phi$ by building a tableau, a proof tree whose root is labelled with this initial sequent. Sequents labelling the immediate successors of a node are determined by an application of one of the rules. The boolean, Box and Diamond rules are straightforward. New constants are introduced when fixed point formulae are met, and then these are unfolded. The Thinning rule allows us to enlarge the set being checked, and is essential for completeness of the system (and need only be applied to sequents with fixed point formulae or constants).

An essential missing ingredient is when a node in a proof tree counts as a leaf. We assume that the rules above only apply to nodes that are not terminal. A node

[1] We assume the interpretation of formulae relative to definition lists as in [11] by in effect treating constants as variables: if Δ is $(U_1 = \Psi_1, \ldots, U_n = \Psi_n)$ then $\|\Phi_\Delta\|^{\mathcal{T}}_{\mathcal{V}} = \|\Phi\|^{\mathcal{T}}_{\mathcal{V}_n}$ where $\mathcal{V}_0 = \mathcal{V}$ and $\mathcal{V}_{i+1} = \mathcal{V}_i[\|\Psi_{i+1}\|^{\mathcal{T}}_{\mathcal{V}_i}/U_{i+1}]$.

n labelled by the sequent $\mathcal{F} \vdash_\Delta \Psi$ is *terminal* if one of the following conditions holds:

(i) $\mathcal{F} = \varnothing$

(ii) $\Psi = Z$ or $\Psi = \neg Z$

(iii) $\Psi = \langle K \rangle \Phi$ and $\exists\, F \in \mathcal{F}\,.\, not(F \xrightarrow{K})$

(iv) $\Psi = U$ and $\Delta(U) = \sigma Z.\Phi$ and there is a node above n in the proof tree labelled $\mathcal{E} \vdash_{\Delta'} U$ with $\mathcal{E} \supseteq \mathcal{F}$

A nodel fulfilling condition (iv) is called a σ-terminal. A node obeying conditions (i) or (iv) when $\sigma = \nu$ is called a *successful* terminal, whereas a node only fulfilling (iii) is *unsuccessful.* In the case of (ii) success depends on the valuation $\mathcal{V}$ of the model: if $\mathcal{F} \subseteq \mathcal{V}(Z)$ when Ψ is Z, or $\mathcal{F} \subseteq \mathcal{P} - \mathcal{V}(Z)$ when Ψ is $\neg Z$ then it is successful, otherwise it is unsuccessful. The definition of a successful μ-terminal, a node complying with (iv) when $\sigma = \mu$, is more complex and requires a little notation.

Suppose a node n is labelled by $\mathcal{E} \vdash_\Delta \Phi$ and n' is an immediate successor of n labelled $\mathcal{E}' \vdash_{\Delta'} \Phi'$. We say that $E' \in \mathcal{E}'$ at n' is a *dependant* of $E \in \mathcal{E}$ at n if

- the rule applied to n is (And), (Or), (Constant Introduction), (Constant Unfolding) or (Thinning), and $E = E'$, or
- the rule is (Box) and $E \xrightarrow{K} E'$, or
- the rule is (Diamond), and $E' = f(E)$.

Assume that the *companion* of a σ-terminal is that node above it which makes it a terminal. Next we define a *trail* to F at a μ-terminal n from E at its companion m to be a sequence of pairs of nodes and processes $(n_1, E_1), \ldots, (n_k, E_k)$ with $(n_1, E_1) = (m, E)$ and $(n_k, E_k) = (n, F)$, such that for all i with $1 \leq i < k$ either

(i) E_{i+1} at n_{i+1} is a dependant of E_i at n_i, or

(ii) n_i is the immediate predecessor of a σ-terminal node n' (where $n' \neq n$) whose companion is n_j for some $j \leq i$, and $n_{i+1} = n_j$ and E_{i+1} at n' is a dependant of E_i at n_i.

Then each companion node n of a μ-terminal induces a pre-order $\sqsupset_n$ by $E \sqsupset_n F$ if there is a trail to F at n from E at its companion. A μ-terminal node n is successful if the ordering $\sqsubset_n$ induced by its companion n is well-founded, that is, there is no infinite descending chain

$$E_0 \sqsupset_n E_1 \sqsupset_n \cdots$$

Finally we say that a tableau is *successful* if it is finite and all its leaves are successful terminals. The tableau technique is both *sound* and *complete* for arbitrary (infinite) transition systems.

Theorem. *$\mathcal{E} \vdash_\Delta \Phi$ is the root of some successful tableau on $(\mathcal{T}, \mathcal{V})$ iff $\mathcal{E} \subseteq \|\Phi_\Delta\|^{\mathcal{T}}_{\mathcal{V}}$*

A full proof of this theorem is given in [2]. The proof is very intricate and directly extends the techniques used in [11]. An interesting question is if the elegant semantic account of local model checking presented in [4, 14] can be extended to cover infinite models.

This tableau method can therefore be used to verify arbitrary temporal properties of arbitrary concurrent systems on arbitrary data (such as Petri Nets, CCS and CSP processes, and concurrent while programs). As the technique is data independent it generalizes standard methods used in program logics which employ induction or appeal to reasoning on well founded structures. Despite its generality we now show that it is applicable to proving properties of processes.

5 Examples

In this section we demonstrate the tableau system on the CCS examples of section 2. Recall the level crossing. Its safety property was expressed as $\Phi = \nu Z.([tcross]ff \lor [ccross]ff) \land [-]Z$. The successful tableau for this property of the crossing (where () is the empty definition list) is:

$$\dfrac{\{Crossing\} \vdash_{()} \Phi}{\dfrac{\dfrac{\mathcal{E} \vdash_{()} \Phi}{\mathcal{E} \vdash_\Delta U}\ {\scriptstyle \Delta=(U=\Phi)}}{\dfrac{\mathcal{E} \vdash_\Delta ([tcross]ff \lor [ccross]ff) \land [-]U}{\dfrac{\mathcal{E} \vdash_\Delta [tcross]ff \lor [ccross]ff}{\dfrac{\mathcal{E}_1 \vdash_\Delta [tcross]ff}{\varnothing \vdash_\Delta ff} \quad \dfrac{\mathcal{E}_2 \vdash_\Delta [ccross]ff}{\varnothing \vdash_\Delta ff}} \qquad \dfrac{\mathcal{E} \vdash_\Delta [-]U}{\mathcal{E} \vdash_\Delta U}}}}$$

where $\mathcal{E}_1$ is all derivatives of $Crossing$ such that the $Rail$ component is not in the state $tcross.\overline{red}.Rail$, and $\mathcal{E}_2$ is all derivatives such that the $Road$ component is not in the state $ccross.\overline{down}.Road$, and $\mathcal{E} = \mathcal{E}_1 \cup \mathcal{E}_2$. By brute force, one checks that $\mathcal{E}$ is closed under $\stackrel{-}{\longrightarrow}$, thus giving the success of the terminal $\mathcal{E} \vdash_\Delta U$.

Recall the fair liveness property for the crossing given by the formula Φ:

$$\begin{array}{lcl} \Phi & = & \nu Z.[car]\Phi_1 \land [-]Z \\ \Phi_1 & = & \mu X.\nu Y_1.(Q \lor [-ccross](\nu Y_2.(R \lor X) \land [-ccross]Y_2)) \land [-ccross]Y_1 \end{array}$$

where Q holds when the crossing is in any state where $Road$ can by itself perform up and R holds in any state where $Rail$ can by itself perform $green$. A proof that the crossing has this property is given by the following tableau where $[-cc]$

abbreviates $[-ccross]$:

$$
\dfrac{\{Crossing\} \vdash_{()} \Phi}{\dfrac{\mathcal{E} \vdash_{()} \Phi}{\dfrac{\mathcal{E} \vdash_{\Delta_1} W}{\dfrac{\mathcal{E} \vdash_{\Delta_1} [car]\Phi_1 \wedge [-]W}{\dfrac{\mathcal{E} \vdash_{\Delta_1} [car]\Phi_1}{\dfrac{\mathcal{E}' \vdash_{\Delta_1} \Phi_1}{\dfrac{\mathcal{E}_1 \vdash_{\Delta_1} \Phi_1}{\dfrac{\mathcal{E}_1 \vdash_{\Delta_2} U}{\dfrac{\mathcal{E}_1 \vdash_{\Delta_2} \nu Y_1.(Q \vee [-cc](\nu Y_2.(R \vee U) \wedge [-cc]Y_2)) \wedge [-cc]Y_1}{\dfrac{\mathcal{E}_1 \vdash_{\Delta_3} V_1}{\dfrac{\mathcal{E}_1 \vdash_{\Delta_3} (Q \vee [-cc](\nu Y_2.(R \vee U) \wedge [-cc]Y_2)) \wedge [-cc]V_1}{\dfrac{\mathcal{E}_1 \vdash_{\Delta_3} Q \vee [-cc]\nu Y_2.(R \vee U) \wedge [-cc]Y_2}{\mathcal{E}_1 - \mathcal{E}_2 \vdash_{\Delta_3} Q \qquad \dfrac{\mathcal{E}_2 \vdash_{\Delta_3} [-cc]\nu Y_2.(R \vee U) \wedge [-cc]Y_2}{\dfrac{\mathcal{E}_3 \vdash_{\Delta_3} \nu Y_2.(R \vee U) \wedge [-cc]Y_2}{\dfrac{\mathcal{E}_3 \vdash_{\Delta_4} V_2}{\dfrac{\mathcal{E}_3 \vdash_{\Delta_4} (R \vee U) \wedge [-cc]V_2}{\dfrac{\mathcal{E}_3 \vdash_{\Delta_4} R \vee U}{\mathcal{E}_3 \vdash_{\Delta_4} R \quad \varnothing \vdash_{\Delta_4} U} \qquad \dfrac{\mathcal{E}_3 \vdash_{\Delta_4} [-cc]V_2}{\varnothing \vdash_{\Delta_4} V_2}}}}}} \qquad \dfrac{\mathcal{E}_1 \vdash_{\Delta_3} [-cc]V_1}{\mathcal{E}'_1 \vdash_{\Delta_3} V_1}}}}}}}} \qquad \dfrac{\mathcal{E} \vdash_{\Delta_1} [-]W}{\mathcal{E} \vdash_{\Delta_1} W}}}}}
$$

The definition lists appealed to in this tableau are as follows:

$$
\begin{array}{lcl}
\Delta_1 & = & (W = \Phi) \\
\Delta_2 & = & \Delta_1 \cdot U = \Phi_1 \\
\Delta_3 & = & \Delta_2 \cdot V_1 = \nu Y_1.(Q \vee [-ccross](\nu Y_2.(R \vee U) \wedge [-ccross]Y_2)) \wedge [-ccross]Y_1 \\
\Delta_4 & = & \Delta_3 \cdot V_2 = \nu Y_2.(R \vee U) \wedge [-ccross]Y_2
\end{array}
$$

The set $\mathcal{E}$ consists of all possible states of the crossing. $\mathcal{E}'$ contains all the configurations in which *car* has just happened, and this is then thinned to $\mathcal{E}_1$ by closing under derivation by non-*ccross* actions; $\mathcal{E}'_1$ is the set of non-*ccross* derivatives of $\mathcal{E}_1$ (and is therefore contained in $\mathcal{E}_1$); $\mathcal{E}_2$ is that subset of $\mathcal{E}_1$ failing the property Q; and $\mathcal{E}_3$ consists of the appropriate successors of $\mathcal{E}_2$. The success of this tableau is easily established: the inclusion conditions hold, and $\sqsubset$ relation for the least fixed point is well-founded because it is empty!

A simple demonstration of how the technique works for infinite state spaces is given by showing that the *Ticker* from section 2 has the property $\mu Z.[\![tick]\!]Z$, that its observable behaviour is just to perform a *finite* number of ticks.

$$
\begin{array}{c}
\{Ticker\} \vdash_{()} \mu Z.[\![tick]\!]Z \\ \hline
\{\, Up_i, Down_j : i \geq 0, j \geq 0 \,\} \vdash_{()} \mu Z.[\![tick]\!]Z \\ \hline
{}_1\,\{\, Up_i, Down_j : i \geq 0, j \geq 0 \,\} \vdash_{\Delta} U \\ \hline
{}_2\,\{\, Up_i, Down_j : i \geq 0, j \geq 0 \,\} \vdash_{\Delta} [\![tick]\!]U \\ \hline
{}_3\,\{\, Down_j : j \geq 0 \,\} \vdash_{\Delta} U
\end{array}
\quad \Delta{=}(U{=}\mu Z.[\![tick]\!]Z)
$$

For this tableau to be successful, we have to show that the μ-terminal leaf node (3) is successful. It clearly satisfies the inclusion condition. Recall that we have to show that $\sqsubset_{(1)}$ is well-founded, and that in this case $E \sqsupset_{(1)} F$ if there is a trail from E at (1) (the companion of (3)) to F at (3). Since there are no other μ-terminals, this is true exactly if there is a finite sequence of processes E_i $(0 \leq i \leq n,\ n \geq 0)$ such that $E = E_0 \stackrel{tick}{\Longrightarrow} E_1 \stackrel{tick}{\Longrightarrow} \cdots \stackrel{tick}{\Longrightarrow} E_n = F$. But since it can be seen from the process definitions that $E \stackrel{tick}{\Longrightarrow} F$ just in case either $E = Up_i$ and $F = Down_j$ for $i, j \geq 0$, or $E = Down_i$ and $F = Down_{i-1}$ for $i \geq 1$, (which can be proved formally by induction on i and j), $\sqsubset_{(1)}$ is well-founded.

We noted that $Ticker$ satisfies the stronger property $[\![\,]\!]\mu Z.[tick](Z \wedge [-tick]ff)$, i.e. that although it may diverge initially, once it starts ticking, it does nothing else until it stops. We can prove this by using the system with $\stackrel{K}{\longrightarrow}$, and expanding the abbreviated modality. The formula becomes

$$\nu Y.(\mu Z.[tick](Z \wedge [-tick]ff)) \wedge [\tau]Y$$

with the following tableau, where $\Psi = \mu Z.[tick](Z \wedge [-tick]ff)$:

$$
\begin{array}{c}
\{Ticker\} \vdash_{()} \nu Y.\Psi \wedge [\tau]Y \\ \hline
\{Up_i, Down_j\} \vdash_{()} \nu Y.\Psi \wedge [\tau]Y \\ \hline
\{Up_i, Down_j\} \vdash_{\Delta} U \\ \hline
\{Up_i, Down_j\} \vdash_{\Delta} \Psi \wedge [\tau]U \quad \Delta{=}(U{=}\nu Y.\Psi{\wedge}[\tau]Y) \\ \hline
\begin{array}[t]{c}
\{Up_i, Down_j\} \vdash_{\Delta} \Psi \\ \hline
{}_1\,\{Up_i, Down_j\} \vdash_{\Delta'} V \quad \Delta'{=}\Delta{\cdot}V{=}\Psi \\ \hline
{}_2\,\{Up_i, Down_j\} \vdash_{\Delta'} [tick](V \wedge [-tick]ff) \\ \hline
{}_3\,\{\, Down_j : j \geq 0 \,\} \vdash_{\Delta'} V \wedge [-tick]ff \\ \hline
\begin{array}[t]{c} {}_4\,\{\, Down_j : j \geq 0 \,\} \vdash_{\Delta'} V \end{array}
\quad
\begin{array}[t]{c} \{\, Down_j : j \geq 0 \,\} \vdash_{\Delta'} [-tick]ff \\ \hline \varnothing \vdash_{\Delta'} ff \end{array}
\end{array}
\qquad
\begin{array}[t]{c}
\{Up_i, Down_j\} \vdash_{\Delta} [\tau]U \\ \hline
\{\, Up_i : i \geq 1 \,\} \vdash_{\Delta} U
\end{array}
\end{array}
$$

Most rule applications here are straightforward. Again, to show that (4) is a successful μ-terminal, note that there is a trail from E at (1) to F at (4) just in case $E = E_0 \xrightarrow{tick} \cdots \xrightarrow{tick} E_n = F$ $(n \geq 1)$, and since $E \xrightarrow{tick} F$ iff $E = Down_i$ and $F = Down_{i-1}$, $\sqsubset_{(1)}$ is well-founded.

The slot machine is infinite state, like $Ticker$, but also has a relatively large state space modulo an integer parameter, so as with the $Crossing$ we shall leave it to the enthusiastic reader to verify our statements about the derivatives. The safety property, that the machine never pays out more than it has in its bank, has the following tableau where Q_0 indicates that the slot machine owes money:

$$\dfrac{\{\, SM_n : n \geq 0 \,\} \vdash_{()} \nu Z.\neg Q_0 \wedge [-]Z}{\dfrac{\mathcal{E} \vdash_{()} \nu Z.\neg Q_0 \wedge [-]Z}{\dfrac{\mathcal{E} \vdash_{\Delta} U}{\dfrac{\mathcal{E} \vdash_{\Delta} \neg Q_0 \wedge [-]U}{\mathcal{E} \vdash_{\Delta} \neg Q_0 \qquad \dfrac{\mathcal{E} \vdash_{\Delta} [-]U}{\mathcal{E} \vdash_{\Delta} U}}}} \; {\scriptstyle \Delta = (U = \nu Z.\neg Q_0 \wedge [-]Z)}}$$

As in previous cases we first use the (Thinning) rule to enlarge the set of slot machines to the set $\mathcal{E}$ of derivatives of $\{\, SM_n : n \geq 0 \,\}$: it is merely tedious to check that $\mathcal{E}$ can be partitioned into nine subsets (according to the state of each component: SM_n, $(bank.(lost.\overline{loss}.IO + release(y).\overline{win}(y).IO) \mid B_n \mid D)\backslash K$, etc.), each parametrized by an integer variable (or in some cases, two), and that this set is closed under $\xrightarrow{-}$ and does not include any SM_j for $j < 0$.

The verification that the slot machine has the weak liveness property that a million pounds can be won infinitely often is given by the following successful tableau:

$$\dfrac{\{\, SM_n : n \geq 0 \,\} \vdash_{()} \nu Y.\mu Z.\langle \overline{win}(10^6)\rangle Y \vee \langle - \rangle Z}{\dfrac{\mathcal{E} \vdash_{()} \nu Y.\mu Z.\langle \overline{win}(10^6)\rangle Y \vee \langle - \rangle Z}{\dfrac{\mathcal{E} \vdash_{\Delta} U}{\dfrac{\mathcal{E} \vdash_{\Delta} \mu Z.\langle \overline{win}(10^6)\rangle U \vee \langle - \rangle Z}{\dfrac{{}_1\, \mathcal{E} \vdash_{\Delta'} V}{\dfrac{{}_2\, \mathcal{E} \vdash_{\Delta'} \langle \overline{win}(10^6)\rangle U \vee \langle - \rangle V}{\dfrac{\mathcal{E}_1 \vdash_{\Delta'} \langle \overline{win}(10^6)\rangle U}{\mathcal{E}'_1 \vdash_{\Delta'} U} \qquad \dfrac{{}_3\, \mathcal{E}_2 \vdash_{\Delta'} \langle - \rangle V}{{}_4\, \mathcal{E}'_2 \vdash_{\Delta'} V}}}} \; {\scriptstyle \Delta' = \Delta \cdot V = \mu Z.\langle \overline{win}(10^6)\rangle U \vee \langle - \rangle Z}}} \; {\scriptstyle \Delta = (U = \nu Y.\mu Z.\langle \overline{win}(10^6)\rangle Y \vee \langle - \rangle Z)}}$$

Some explanation is called for here, but again, the details are omitted. As before, $\mathcal{E}$ is the set of all derivatives. The vital rules in this tableau are the disjunction at node (1), where $\mathcal{E}_1$ is exactly those processes capable of performing a $\overline{win}(10^6)$ action, and $\mathcal{E}_2$ is the remainder; and the (Diamond) rule at node (3), where f is defined to ensure that $\mathcal{E}_1$ is eventually reached: for process with less than 10^6 in the bank, f chooses events leading towards loss, so as to increase the amount in the bank; and for processes with more than 10^6, f chooses to $\overline{\mathit{release}}(10^6)$. The formal proof requires partitioning $\mathcal{E}_2$ into several classes, each parametrized by an integer n, and showing that while $n < 10^6$, n is strictly increasing over a cycle through the classes; then when $n = 10^6$, f selects a successor that is not in $\mathcal{E}_2$, and so a chain $E_0 \stackrel{-}{\longrightarrow} \cdots$ through nodes (1), (2), (3), (4) terminates, and therefore $\sqsubset_{(1)}$ is well-founded.

Finally, we consider the network of processes and relay stations. Recall that a weak liveness property is $\mu Z.\Psi \lor \langle - \rangle Z$ where Ψ is the formula $\bigwedge_{1 \leq j < i \leq n} \langle \overline{\mathit{line}}(j,i) \rangle \mathit{tt}$ A successful tableau for this has the form:

$$\frac{\{\text{initial config}\} \vdash_{()} \mu Z.\Psi \lor \langle - \rangle Z}{\dfrac{\mathcal{E}_1 \vdash_{()} \mu Z.\Psi \lor \langle - \rangle Z}{\dfrac{\mathcal{E}_1 \vdash_\Delta U}{\dfrac{\mathcal{E}_1 \vdash_\Delta \Psi \lor \langle - \rangle U}{\dfrac{\mathcal{E}_2 \vdash_\Delta \Psi}{\dfrac{\mathcal{E}_2 \vdash_\Delta \langle \overline{\mathit{line}}(1,2) \rangle \mathit{tt}}{\mathcal{F}_{1,2} \vdash_\Delta \mathit{tt}} \quad \cdots \quad \dfrac{\mathcal{E}_2 \vdash_\Delta \langle \overline{\mathit{line}}(n-1,n) \rangle \mathit{tt}}{\mathcal{F}_{n-1,n} \vdash_\Delta \mathit{tt}}} \qquad \dfrac{\mathcal{E}_3 \vdash_\Delta \langle - \rangle U}{\mathcal{E}_4 \vdash_\Delta U}}}}}$$

We describe, fairly informally, the construction of a successful tableau. Choose a cell owned by P_1 and a cell owned by P_2. Draw a line on the grid from one cell to the other. Starting from the initial configuration, select P_1 to fire the $\overline{\mathit{alloc}}$ action for the chosen cell, and then similarly for P_2; then continue choosing $\overline{\mathit{alloc}}$ actions to allocate the next cell on the line along from P_1. When the line has been followed to the P_2 cell, that cell will be in the state $\overline{\mathit{line}}(1,2).0$.

Repeat for the other pairs of processes, drawing each new line so that it does not cross any earlier line. When all pairs have been connected in this manner, the resulting configuration satisfies Ψ, and forms the sole member of the set $\mathcal{E}_2$; all the intervening configurations form $\mathcal{E}_3$, and $\mathcal{E}_4$ is $\mathcal{E}_3$ less the start state.

Now suppose a successful tableau exists. Follow the $\sqsupset$ relation from the initial state until it terminates, and each time through the diamond rule, draw a line on the grid corresponding to the action chosen: if the action involves the $\overline{\mathit{alloc}}_{i\,j+1}$ action of C_{ij}, draw a line from (i,j) to $(i,j+1)$. The result of this procedure is to draw disjoint lines between each pair of processes, for a $\overline{\mathit{line}}(k,k')$ action is possible only if a line from a P_k cell meets a line from a P'_k cell.

Now, since we stipulated that the cells owned by a given process be adjacent cells on a line, they may be collapsed to a point to give the complete graph on n vertices; since the lines do not cross, the graph is planar; and therefore no successful tableau exists if $n \geq 5$.

This system can be elaborated in many ways. For instance consider a persistent system that reconfigures after, say, a power failure. In this case, one would want to specify that after a $\overline{\mathit{reset}}$ action (which we assume causes all components to revert to their initial state) Ψ is achieved again, and that this cycle continues.

Further, if we adopt the less restrictive requirement that Ψ should be that all $\overline{line}$ actions occur in any order, we have an example of a fairly complex cylic property. These properties can be expressed in the mu-calculus; however, the formulae become quite long. A simple cyclic property is that every run involves the sequence of labels $(ab)^\omega$. This is expressed by the following formula:

$$\nu Z.[-a]ff \wedge \langle - \rangle tt \wedge [-]([-b]ff \wedge \langle - \rangle tt \wedge [-]Z)$$

This illustrates the desirability of providing the user with a suitably chosen set of formula macros to express common properties succintly and, if feasible, to give derived tableau rules for such macros. This is one topic for further investigation: another important topic is the use of compositional reasoning within the system.

References

References

[1] J. Bradfield, 'Proving temporal properties of Petri Nets', *Proc. 11th Int. Conf. on Theory and Application of Petri Nets*, 1990.

[2] J. Bradfield, C. Stirling, 'Local model checking for infinite state spaces', Technical Report LFCS-90-115, University of Edinburgh 1990.

[3] E. M. Clarke, E. A. Emerson, A. P. Sistla, 'Automatic verification of finite-state concurrent systems using temporal logic specifications: a practical approach' *Tenth ACM Symposium on Principles of Programming Languages.* ACM, Austin, Texas, 1983.

[4] R. Cleaveland, 'Tableau-based model checking in the Propositional Mu-Calculus', Technical Report, University of Sussex 1988. (To appear in *Acta Informatica*)

[5] M. Hennessy, R. Milner, 'Algebraic laws for nondeterminism and concurrency', *J. Assoc. Comput. Mach.* 32, 137-161, 1985.

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

[7] K. Larsen, 'Proof systems for Hennessy-Milner logic with recursion', *Proc. CAAP* 1988. (To appear in *Inf. and Comp.*)

[8] R. Milner, *Communication and Concurrency*, Prentice-Hall 1989.

[9] V. Pratt, 'A decidable μ-calculus', *Proc. 22nd. FOCS*, 421–27, 1981.

[10] C. Stirling, 'Temporal logics for CCS', LNCS 354, 660-672, 1989.

[11] C. Stirling, D. J. Walker, 'Local model checking in the modal mu-calculus', *TAPSOFT 89* (369–382). LNCS 351, 1989. (To appear in *Theoretical Computer Science*)

[12] C. Stirling, D. J. Walker, 'A general tableau technique for verifying temporal properties of concurrent programs', *Proc. Int. BCS-FACS Workshop on Semantics For Concurrency*, 1990.

[13] D. J. Walker, 'Automated analysis of mutual exclusion algoritms using CCS', *Formal Aspects of Computing* 1, 273-292, 1989.

[14] G. Winskel, 'Model checking the modal nu-Calculus', *Proc. ICALP* 1989.