Verifying pCTL Model Checking

Johannes Hölzl* and Tobias Nipkow

Institut für Informatik, Technische Universität München
www.in.tum.de/~hoelzl, www.in.tum.de/~nipkow

Abstract. Probabilistic model checkers like PRISM check the satisfiability of probabilistic CTL (pCTL) formulas against discrete-time Markov chains. We prove soundness and completeness of their underlying algorithm in Isabelle/HOL. We define Markov chains given by a transition matrix and formalize the corresponding probability measure on sets of paths. The formalization of pCTL formulas includes unbounded cumulated rewards.

1 Introduction

Modeling systems as discrete-time Markov chains is a popular technique to analyze probabilistic behavior of network protocols, algorithms, communication systems or biological systems. Probabilistic model checkers, like PRISM [13] or MRMC [10], interpret Markov chains and analyze quantitative properties, specified as probabilistic CTL (pCTL) formulas [6]. In this paper we formalize the background theory and the algorithm used by these probabilistic model checkers in the proof assistant Isabelle/HOL [20].

Our (almost) executable model checker is certainly not a rival to any of the existing model checkers. Instead, our work should be seen as a foundational contribution that paves the way towards a fruitful combination of automatic and interactive verification methods. Possible application scenarios include the following: interactive verification of parameterized systems, or a verified checker that checks individual runs of a hand coded model checker that produces a certificate. We discuss these in more detail in the conclusion. Quite apart from applications, we see our work as another building block in the larger undertaking of formalizing key areas of computer science (as is currently happening with compilers [15] and operating system kernels [11]).

This is the first time probabilistic model checking has been formalized in a proof assistant. So far, the necessary mathematical background theories were simply not available. We start from our recent formalization of measure theory in Isabelle/HOL, including the Lebesgue integral and Caratheodory's theorem [8]. Based on this, Section 3 formalizes the following material:

- infinite products of probability spaces,
- Markov chains defined by a transition matrix and the existence of their probability measure on paths,

* Supported by the DFG Graduiertenkolleg 1480 (PUMA).

C. Flanagan and B. König (Eds.): TACAS 2012, LNCS 7214, pp. 347–361, 2012.

– properties of paths reaching a set of states almost everywhere.

Now we have the necessary theory to define, in Section 4, syntax, semantics and a model checking algorithm for pCTL, and verify the algorithm wrt the semantics, following the standard literature [12].

2 Related Work

There are a number of formalizations and proofs of aspects of model checking: verification of a model checker for the modal μ-calculus [23], of partial order reduction [3], of two innermost loops of a model checker for real time CTL [21], of a CTL model checker [20], of a model checker for dynamic pushdown networks [14], and of the translation from LTL to Büchi automata [22]. But none involve probabilities.

The formalization of probability theory in HOL starts with Hurd's thesis [7]. He introduces measure theory, proves Caratheodory's theorem about the existence of measure spaces and uses it to introduce a probability space on infinite boolean sequences. He provides methods to generate discrete random variables with Bernoulli or uniform distribution. Based on this work Liu *et al.* [17] formalize the concept of Markov chains. Their theory does not provide everything we need: it lacks a probability measure on paths, and their measure space needs to be the type universe whereas we relax it to sets. Coble [4] and Mhamdi *et al.* [18] introduce generalized measure spaces on sets, the extended real numbers $\overline{\mathbb{R}}$ and the Lebesgue integral. However, there is no theorem to show the existence of a measure space.

3 Foundations

3.1 Isabelle/HOL

The formalizations presented in this paper are done in the Isabelle/HOL theorem prover. In this section we give an overview of our syntactic conventions.

The term syntax follows the λ-calculus, i.e. function application is juxtaposition as in $f\ t$. The notation $t :: \tau$ means that t has type τ. Types are built from the base types $\mathbb{B}$ (booleans), $\mathbb{N}$ (natural numbers), $\mathbb{R}$ (reals), $\overline{\mathbb{R}} = \mathbb{R} \cup \{\infty, -\infty\}$, and type variables ($\alpha$, β etc) via the function type constructor $\Rightarrow$. In particular, (infinite) sequences have type $\mathbb{N} \Rightarrow \alpha$ and are usually denoted by ω.

Prepending an element x to a sequence ω is written as $x{\cdot}\omega$, i.e. $(x{\cdot}\omega)\ 0 = x$ and $(x{\cdot}\omega)\ (n{+}1) = \omega\ n$. The while-combinator *while* $:: (\alpha \Rightarrow \mathbb{B}) \Rightarrow (\alpha \Rightarrow \alpha) \Rightarrow \alpha \Rightarrow \alpha$ satisfies the standard recursion equation:

$$\textit{while}\ P\ f\ x = \textit{if}\ P\ x\ \textit{then}\ \textit{while}\ P\ f\ (f\ x)\ \textit{else}\ x$$

We write $\bigtimes_{i \in I} A\ i := \{f \mid \forall i \in I.\ f\ i \in A\ i\}$[1] for the dependent function space (which is a set, not a type in HOL); if A is constant we write $I \to A$.

[1] We use $\bigtimes$ for products of sets, and $\prod$ for products of numbers.

To represent non-total functions in HOL we use the option data type

$$\alpha\ option = \mathsf{Some}\ \alpha \mid \mathsf{None}$$

whose values are $\mathsf{Some}\ x$ for $x :: \alpha$ and None. We introduce the option-monad to combine non-total functions to new non-total functions. The infix bind-operator $\gg\!=$ is defined by the equations $((\mathsf{Some}\ x) \gg\!= f) = f\ x$ and $(\mathsf{None} \gg\!= f) = \mathsf{None}$. Notation *return* is equal to Some. Similar to Haskell's monad-syntax we use the *do*-syntax to represent chains of bind-operators, for example:

$$\begin{array}{ll} do & x \leftarrow f \\ & y \leftarrow g\ x \\ & let\ z = h\ x\ y \\ & return\ (x + y + z) \end{array} \quad \Longrightarrow \quad \begin{array}{l} r \gg\!= (\lambda x. \\ \quad g\ x \gg\!= (\lambda y. \\ \quad\quad \mathsf{Some}\ (x + y + h\ x\ y))) \end{array}$$

We use the option-monad not only to represent non-total functions, but also to write the algorithm in a more imperative style. The only non-total function in the pCTL model checking algorithm is Gauss-Jordan elimination.

3.2 Probability Space

Probabilistic CTL formulas are defined in terms of probabilities on sets of paths. To define a probability space on paths we use the measure theory formalized in [8]. This provides us with the concepts of extended real numbers, σ-algebras, measure spaces, the Lebesgue integral, the Lebesgue measure and, as a way to construct measures, Caratheodory's theorem. We write $\mathcal{B}$ for the Borel sets, $\sigma(\mathcal{A})$ for the σ-algebra generated by $\mathcal{A}$, $f \in \mathcal{A}_1 \rightarrow_M \mathcal{A}_2$ for a measurable function f mapping from $\mathcal{A}_1$ to $\mathcal{A}_2$, $\int_\omega f\ \omega d\mu$ for the Lebesgue integral, and $\mathrm{AE}_\mu\, \omega.\ P\ \omega$ if the predicate P holds *almost everywhere*, i.e. the complement of P is a subset of a null set in μ. In the following two sections we will introduce the infinite product of probability spaces and based on this a probability measure on paths in Markov chains. This was not yet formalized in [8], and we are not aware of any formalization of these concepts in interactive theorem provers.

A probability space is a measure space which assigns 1 to the entire space:

Definition 1. *Probability space*

$$\mathsf{prob\text{-}space}\ (\Omega, \mathcal{A}, \mu) :\longleftrightarrow \mathsf{measure\text{-}space}\ (\Omega, \mathcal{A}, \mu) \wedge \mu\ \Omega = 1$$

Our first step in introducing a probability space for paths is to formalize the infinite product of probability spaces $(\Omega\ i, \mathcal{A}\ i, \mu\ i)$, for i in some index set I. We used the proof in [2] as the base of our formalization of infinite products. The space of infinite products of probability spaces is the function space $\Omega_P := \bigtimes_{i \in I} \Omega\ i$. The generating set of infinite products is the collection of all embeddings of finite products:

Definition 2. *Embedding of finite products and the product σ-algebra $\mathcal{A}_P$*

$$\begin{array}{ll} \mathsf{emb}\ J\ F & := \big\{\omega \in \Omega_P \mid \forall i \in J.\ \omega\ i \in F\ i\big\} \\ \mathcal{A}_P & := \sigma\big(\big\{\mathsf{emb}\ J\ F \mid \mathsf{finite}\ J \wedge J \subseteq I \wedge (\forall i \in J.\ F\ i \in \mathcal{A}\ i)\big\}\big) \end{array}$$

With Caratheodory's theorem we show that a probability measure on $\mathcal{A}_P$ exists which maps $\mathsf{emb}\ J\ F$ to the product of the real numbers $\mu\ i\ (F\ i)$, the property we want to have from a product space.

Theorem 3. *Probability measure on $\mathcal{A}_P$*
There exists a unique probability measure μ_P on $\mathcal{A}_P$

$$\textsf{prob-space}\ (\Omega_P, \mathcal{A}_P, \mu_P)$$

with: If $J \subseteq I$ is finite and $F \in \bigtimes_{i\in J} \mathcal{A}\ i$ then

$$\mu_P(\mathsf{emb}\ J\ F) = \prod_{i\in J} \mu\ i\ (F\ i)\ .$$

We choose such a probability measure μ_P with $I := \mathbb{N}$ and $\mathcal{A}\ i := \lambda_{[0;1[}$, the Borel-Lebesgue measure restricted to $[0;1[$. Hence μ_P is now a probability measure on sequences $\mathbb{N} \to [0;1[$. From the equation in Theorem 3 we have

$$\mu_P\Big(\bigtimes_i F\ i\Big) = \prod_i \lambda_{[0;1[}\ (F\ i)\ .$$

Hence the elements in the product space induce countably many, independent random variables with a continuous, uniform distribution. The formalization in [7] only provides a probability measure on sequences $\mathbb{N} \to \mathbb{B}$, which induces random variables with a discrete distribution.

3.3 Markov Chains

We introduce Markov chains as probabilistic automata, i.e. as discrete-time time-homogeneous finite-space Markov processes. A Markov chain is defined by its state space S and an associated transition matrix τ. We assume no initial distribution or starting state, however when measuring paths we always provide a starting state. A path on a Markov chain is a function $\mathbb{N} \to S$, i.e. an infinite sequence of states visited in the Markov chain.

Definition 4. *Markov chain*

$$\begin{aligned}\textsf{markov-chain}\ S\ \tau :\longleftrightarrow\ & \textsf{finite}\ S \wedge S \neq \emptyset \\ & \wedge\ \forall s, s' \in S.\ 0 \leq \tau\ s\ s' \\ & \wedge\ \forall s \in S.\ \big(\textstyle\sum_{s'\in S} \tau\ s\ s'\big) = 1\end{aligned}$$

For the rest of the paper we assume a Markov chain with state space S and transition matrix τ. We write $E(s)$ for the set of all successor states, i.e. all $s' \in S$ with $\tau\ s\ s' \neq 0$. Note that a path ω does not require that $\omega\ (i+1)$ is a successor of $\omega\ i$. Our first goal is to define a probability space $(\Omega, \mathcal{T}, \mu_s)$ on the space of all paths $\mathbb{N} \to S$. We call the set of all paths starting with a common prefix, namely ω', $\mathsf{Cy}\ \omega'\ n := \{\omega \in \mathbb{N} \to S \mid \forall i < n.\ \omega\ i = \omega'\ i\}$, a *cylinder*. The probability measure on paths assigns to cylinders the product of the transition probabilities:

Definition 5. *Path σ-algebra, and pre-measure μ'_P*

$$\begin{array}{rll} & \Omega & := \mathbb{N} \to S \\ & \mathcal{T} & := \sigma(\{\mathsf{Cy}\ \omega\ n \mid \omega \in \Omega\}) \\ \forall \omega \in \mathbb{N} \to S, s \in S, n. & \mu'_s\ (\mathsf{Cy}\ \omega\ n) & := \prod_{i<n} \tau\ ((s \cdot \omega)\ i)\ (\omega\ i) \end{array}$$

Note that μ'_s explicitly carries the starting state, hence we assign to $\mathsf{Cy}\ \omega\ n$ the transition probability for the steps $s \to_\tau \omega\ 0 \to_\tau \omega\ 1 \to_\tau \cdots \to_\tau \omega\ (n-1)$. Before we use this as a probability space we need to show that μ'_s can be extended to a probability measure. To this end, we provide a function *path* which constructs a path out of a sequence $\mathbb{N} \to [0;1[$, and show that this function is measurable.

As S is finite and not empty we know that there exists a bijective function mapping from $\{0, \dots, |S|-1\}$ to S, we define *order* to be such a function. Using *order* we introduce *select* which splits $[0;1[$ into disjoints intervals of size $\tau\ s\ s'$, see Fig. 1. The recursive function *path* now walks along a sequence X of values in the unit interval and selects the next state.

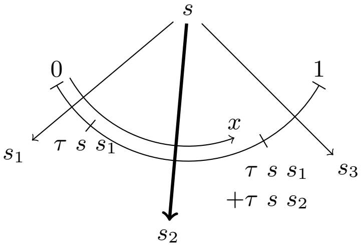

Fig. 1. The next state after s selected by x is $s_2 = \mathsf{select}\ s\ x$

Definition 6. *Path selection*

$$\begin{array}{ll} \mathsf{select}\ s\ x & := \mathsf{order}\ \left(\min\left\{i \mid x < \sum_{j \le i} \tau\ s\ (\mathsf{order}\ j)\right\}\right) \\ \mathsf{path}\ s\ X\ 0 & := \mathsf{select}\ s\ (X\ 0) \\ \mathsf{path}\ s\ X\ (n+1) & := \mathsf{select}\ (\mathsf{path}\ s\ X\ n)\ (X\ (n+1)) \end{array}$$

The set $T\ s\ s' := \{x \in [0;1[\mid \mathsf{select}\ s\ x = s'\}$ is measurable and $\lambda_{[0;1[}(T\ s\ s') = \tau\ s\ s'$. We represent the inverse image of cylinders over *path* with *emb* and T:

Lemma 7. *For all states s, paths ω, and prefix length n:*

$$\left\{\omega \in \Omega_P \mid \mathsf{path}\ s\ \omega \in \mathsf{Cy}\ \omega\ n\right\} = \mathsf{emb}\ \{0, 1, \dots, n-1\}\ (\lambda i.\ T\ ((s \cdot \omega)\ i)\ (\omega\ i))$$

As $\mathsf{Cy}\ \omega\ n$ are the generators of $\mathcal{T}$ and *emb* is measurable in $\mathcal{A}_P$, $\mathsf{path}\ s$ is in $\mathcal{A}_P \to_M \mathcal{T}$. With this we show that $\mu_s\ A := \mu_P\left\{\omega \in \Omega_P \mid \mathsf{path}\ s\ \omega \in A\right\}$ defines a probability measure, and with Lemma 7 we show that μ_P extends μ'_P.

Theorem 8. *μ_s is the unique probability measure on paths which extends μ'_s*

$$\textsf{prob-space}\ (\Omega, \mathcal{T}, \mu_s)$$

$$\forall \omega \in \Omega, s \in S, n.\ \mu_s\ (\textsf{Cy}\ \omega\ n) = \textstyle\prod_{i<n} \tau\ ((s \cdot \omega)\ i)\ (\omega\ i)$$

The Markov chain induces *iterative equations* on the measure μ_s, the Lebesgue integral and the Æ-quantifier, relating properties about s to properties of $E(s)$, states that are not successors of s are ignored. These equations are often useful in inductive proofs, and already give a hint how to implement a probabilistic model checker. In the rest of the paper we write the Æ-quantifier on the path measure μ_s as $\text{Æ}_s\,\omega.\ P\ \omega$ instead of $\text{Æ}_{\mu_s}\,\omega.\ P\ \omega$.

Theorem 9. *Iterative equations for μ_s, the Lebesgue integral and Æ_s*

If s, A, P, and f are measurable, i.e. s is in S, A and $\{\omega \in \Omega \mid P\ \omega\}$ are in $\mathcal{T}$, and f is in $\mathcal{T} \rightarrow_M \mathcal{B}$ then the following equations hold:

$$\begin{aligned}
\mu_s\ A &= \sum_{s' \in E(s)} \tau\ s\ s' \cdot \mu_{s'} \{\omega \in \Omega \mid s' \cdot \omega \in A\} \\
\int_\omega f\ \omega d\mu_s &= \sum_{s' \in E(s)} \tau\ s\ s' \cdot \int_\omega f\ (s' \cdot \omega) d\mu_{s'} \\
\text{Æ}_s\,\omega.\ P\ \omega &\longleftrightarrow \forall s' \in E(s).\ \text{Æ}_{s'}\,\omega.\ P\ (s' \cdot \omega)
\end{aligned}$$

We prove the iterative equation for μ_s by proving the equality when A is a cylinder, with the uniqueness of measures [8] follows that they are equal for all measurable sets A. Based on this the integral equation is shown for simple functions, and then for $\mathcal{B}$-measurable functions.

A state s' is reachable in Φ starting in s iff there is a non-zero probability to reach s' by only going through the specific set of states Φ. The starting state s and the final state s' are not necessary in Φ.

Definition 10. *Reachability of states*

$$\begin{aligned}
\textsf{reachable}\ \Phi\ s := \{s' \in S \mid \exists \omega \in \Omega, n.\ &(\forall i \le n.\ \omega\ i \in E((s \cdot \omega)\ i)) \wedge \\
&(\forall i < n.\ \omega\ i \in \Phi) \wedge \omega\ n = s'\}
\end{aligned}$$

Reachability is a purely qualitative property, as it is defined on the graph of non-zero transitions. Hence an upper bound R of $\textsf{reachable}\ \Phi\ s$ is given when all successor states of $R \cap \Phi$ are in R again.

Lemma 11. *Sets closed under E contain* $\textsf{reachable}$

$$\begin{aligned}
&s \in R \cap \Phi \wedge (\forall t \in R \cap \Phi.\ E(t) \subseteq R) \wedge R \subseteq S \wedge \Phi \subseteq S \\
&\longrightarrow \textsf{reachable}\ \Phi\ s \subseteq R
\end{aligned}$$

The until-operator introduces a similar concept on paths. Its definition does not assume that a state is a successor state of the previous one, as this is already ensured by the probability measure μ_s.

Definition 12. *Until on paths*

$$\mathit{until}\ \Phi\ \Psi := \{\omega \in \Omega \mid \exists n.\ (\forall i < n.\ \omega\ i \in \Phi) \wedge \omega\ n \in \Psi\}$$

Can we compute the probability of $\mu_s(\mathit{until}\ \Phi\ \Psi)$ by only using *reachable*? It is easy to show that $\mu_s(\mathit{until}\ \Phi\ \Psi) = 0$ iff $(\mathit{reachable}\ \Phi\ s) \cap \Psi = \emptyset$. But is there also a method to characterize $\mu_s(\mathit{until}\ \Phi\ \Psi) = 1$ in terms of *reachable*? For this we need to introduce state fairness. A path ω is *state fair* w.r.t. s and t if t appears infinitely often as the successor of s in ω, provided that s appears infinitely often. The definition and proofs about state fairness are based on Baier [1].

Definition 13. *State fairness*

$$\begin{aligned}&\mathit{fair}\ s\ t := \\ &\quad \{\omega \in \Omega \mid (\exists n.\ \forall i \geq n.\ \omega\ i \neq s) \vee (\forall n.\ \exists i \geq n.\ \omega\ i = s \wedge \omega\ (i+1) = t)\}\end{aligned}$$

Baier [1] defines state fairness and a more general version called p-fairness, but we only need state fairness. We show that almost every path is state fair for each state and its successors.

Lemma 14. *Almost every path is state fair*

$$\forall s \in S.\ \mathit{AE}_s\,\omega.\ \forall s' \in S.\ \forall t' \in E(s').\ s{\cdot}\omega \in \mathit{fair}\ s'\ t'$$

Using this we prove that starting in a state s almost every path fulfills $\mathit{until}\ \Phi\ \Psi$ if (1) all states reachable by Φ are in Φ or Ψ and (2) each state reachable from s has again the possibility to reach Ψ. This theorem allows us to prove that $\mathit{until}\ \Phi\ \Psi$ holds almost everywhere by a reachability analysis on the graph, and hence $\mu_s(\mathit{until}\ \Phi\ \Psi) = 1$.

Theorem 15. *Reachability implies until*

$$\begin{aligned}&s \in \Phi \wedge \Phi \subseteq S \wedge \mathit{reachable}\ (\Phi \setminus \Psi)\ s \subseteq \Phi \cup \Psi \\ &\wedge \forall t \in (\mathit{reachable}\ (\Phi \setminus \Psi)\ s \cup \{s\}) \setminus \Psi.\ \mathit{reachable}\ (\Phi \setminus \Psi)\ t \cap \Psi \neq \emptyset \\ &\longrightarrow \mathit{AE}_s\,\omega.\ s{\cdot}\omega \in \mathit{until}\ \Phi\ \Psi\end{aligned}$$

The hitting time on a path ω is the first index at which a state from a set Φ occurs.

Definition 16. $\mathit{hitting\text{-}time}\ \Phi\ \omega = \min\{i \mid \omega\ i \in \Phi\}$

For the computation of rewards it is important to know if the expected hitting time is finite. Standard textbook proofs assume an irreducible chain. We took such a proof from [16], and adapted it to our setting. Instead of a irreducible chain we assume Φ is always reached from s. We show that the expected hitting time of Φ for paths starting in s is finite if almost every path starting in s reaches Φ.

Theorem 17. *Finite expected hitting time*
If s is in S and $\mathit{AE}_s\,\omega.\ s{\cdot}\omega \in \mathit{until}\ S\ \Phi$ then

$$\int_\omega \mathit{hitting\text{-}time}\ \Phi\ (s{\cdot}\omega) d\mu_s \neq \infty$$

4 Verifying pCTL Model Checking

4.1 pCTL Formulas

We do not introduce a labeled Markov chain as [12] does, instead we define labels to be subsets of S. We introduce a Markov chain with rewards as a Markov chain with ρ, the rewards associated per state, and ι, the rewards associated per transitions. These rewards are non-negative, real numbers.

Definition 18. *Markov chain with rewards*

$$\begin{aligned}\textsf{rewarded-markov-chain}\ S\ \tau\ \rho\ \iota := {} & \textsf{markov-chain}\ S\ \tau \\ & \wedge \forall s \in S.\ 0 \le \rho\ s \\ & \wedge \forall s, s' \in S.\ 0 \le \iota\ s\ s'\end{aligned}$$

For the rest of the paper we assume a Markov chain with rewards, with the state space S, the transition matrix τ, and the reward functions ρ and ι.

The pCTL syntax is introduced as an inductive data type.

Definition 19. *pCTL syntax*

$$\begin{aligned}\textsf{sform} &:= \textsf{label}\ \mathcal{P}(S) \mid \neg\textsf{sform} \mid \textsf{sform} \wedge \textsf{sform} \\ &\mid P^{\bowtie\mathbb{R}}\ \textsf{pform} \mid E^{\bowtie\mathbb{R}}\ \textsf{eform} \\ \textsf{pform} &:= X\ \textsf{sform} \mid \textsf{sform}\ U^{\le\mathbb{N}}\ \textsf{sform} \mid \textsf{sform}\ U^{\infty}\ \textsf{sform} \\ \textsf{eform} &:= C^{<\mathbb{N}} \mid I^{=\mathbb{N}} \mid F^{\infty}\ \textsf{sform} \\ \bowtie &:= \le \mid < \mid = \mid > \mid \ge\end{aligned}$$

Informally, a state s fulfills $P^{\bowtie r}\ \Phi$ (or $E^{\bowtie r}\ \Phi$) if the probability (expected reward) of the paths starting in s and fulfilling Φ is related with $\bowtie r$. A path fulfills $X\ \Phi$ if its second state fulfills Φ. A path fulfills $\Phi\ U^{\le k}\ \Psi$ (or $\Phi\ U^{\infty}\ \Psi$, the unbounded until) if it stays in Φ, until it reaches Ψ in at least k steps (at some step). The reward $C^{<k}$ sums all state and transitions rewards for the first k steps, $I^{=k}$ is the state reward at step k, and the unbounded cumulated reward $F^{\infty}\ \Phi$ sums rewards until Φ is reached, if it is never reached it is infinity. We define now semantics to assign a formal meaning to the pCTL syntax, cf. [6,12].

Definition 20. *pCTL semantics*

$$\begin{aligned}[\![\textsf{label}\ S']\!] &:= \{s \in S \mid s \in S'\} \\ [\![\neg\ \Phi]\!] &:= S \setminus [\![\Phi]\!] \\ [\![\Phi\ \wedge\ \Psi]\!] &:= [\![\Phi]\!] \cap [\![\Psi]\!] \\ [\![P^{\bowtie r}\ \Phi]\!] &:= \{s \in S.\ \mu_s\{\omega \in \Omega \mid [\![\Phi, s{\cdot}\omega]\!]_P\} \bowtie r\} \\ [\![E^{\bowtie r}\ \Phi]\!] &:= \{s \in S \mid \textstyle\int_\omega [\![\Phi, s{\cdot}\omega]\!]_E d\mu_s \bowtie r\}\end{aligned}$$

$$\begin{aligned}[\![X\ \Phi, \omega]\!]_P &:\longleftrightarrow \omega\ 1 \in [\![\Phi]\!] \\ [\![\Phi\ U^{\le k}\ \Psi, \omega]\!]_P &:\longleftrightarrow \exists n \le k.\ \omega\ n \in [\![\Psi]\!] \wedge (\forall i < n.\ \omega\ i \in [\![\Phi]\!]) \\ [\![\Phi\ U^{\infty}\ \Psi, \omega]\!]_P &:\longleftrightarrow \exists n.\ \omega\ n \in [\![\Psi]\!] \wedge (\forall i < n.\ \omega\ i \in [\![\Phi]\!])\end{aligned}$$

$$\begin{aligned}[\![C^{<k}, \omega]\!]_E &:= \textstyle\sum_{i<k} \rho\ (\omega\ i) + \iota\ (\omega\ i)\ (\omega\ (i+1)) \\ [\![I^{=k}, \omega]\!]_E &:= \rho\ (\omega\ k) \\ [\![F^{\infty}\ \Phi, \omega]\!]_E &:= \begin{cases} [\![C^{<\textsf{hitting-time}\ [\![\Phi]\!]\ \omega}, \omega]\!]_E & \textit{if}\quad \exists i.\ \omega\ i \in [\![\Phi]\!] \\ \infty & \textit{otherwise}\end{cases}\end{aligned}$$

We see that $\llbracket \Phi \rrbracket$ is a subset of S and hence also finite. The set $\{\omega \in \Omega \mid \llbracket \Phi, \omega \rrbracket_P\}$ is measurable in $\mathcal{T}$, and $\lambda\omega.\ \llbracket \Phi, \omega \rrbracket_E \in \mathcal{T} \rightarrow_M \mathcal{B}$, i.e. is Borel-measurable on $\mathcal{T}$. So the probability for $\llbracket P^{\bowtie r}\ \Phi \rrbracket$, and the integral for $\llbracket E^{\bowtie r}\ \Phi \rrbracket$ are well-defined.

4.2 Verifying the Algorithm

The model checking algorithm *Sat* for pCTL formulas is based on three methods:

- Iterative methods to compute the probability of bounded until and the expectation of bounded rewards
- Reachability analysis on the graph of non-zero transitions to compute the sets $\llbracket P^{=0}(\Phi\ U^\infty\ \Psi) \rrbracket$ and $\llbracket P^{=1}(\Phi\ U^\infty\ \Psi) \rrbracket$.
- Solving systems of linear equations for the unbounded until operator and unbounded rewards. This requires the previous methods to construct a system of linear equations with a unique solution.

Solving systems of linear equations may (in general) fail. To cater for this possibility we use option values in our computation and formulate our algorithm with the help of the *do*-syntax (recall Section 3.1).

The definition and the correctness proof of the algorithm *Sat* is by induction over the syntax of pCTL formulas. For a better overview of the formalization we split the definition of *Sat* into multiple parts interleaved with the necessary auxiliary definitions. The final soundness theorem states that *Sat* Φ returns a result and computes the set of states s for which $s \in \llbracket \Phi \rrbracket$ holds, i.e. *Sat* $\Phi =$ *Some* $\llbracket \Phi \rrbracket$.

The definition of *Sat* on *label* S', $\neg\Phi$, $\Phi \wedge \Psi$, and $P^{\bowtie r}(X\Phi)$ is easy. The soundness proof of the first three is done automatically, the last one needs Theorem 9.

Definition 21. *Computing pCTL-satisfiability (1)*

```
Sat (label $S'$)              := return $\{s \in S \mid s \in S'\}$
Sat ($\neg\ \Phi$)                := do
                                $F \leftarrow$ Sat $\Phi$
                                return $(S \setminus F)$
Sat ($\Phi\ \wedge\ \Psi$)              := do
                                $F_1 \leftarrow$ Sat $\Phi$
                                $F_2 \leftarrow$ Sat $\Psi$
                                return $(F_1 \cap F_2)$
Sat ($P^{\bowtie r}\ (X\ \Phi)$)  := do
                                $F \leftarrow$ Sat $\Phi$
                                return $\left\{s \in S \mid \left(\sum_{s' \in F} \tau\ s\ s'\right) \bowtie r\right\}$
```

The iterative methods to compute bounded until (*ProbUb* $k\ s\ S_1\ S_2$), cumulative expectation (*ExpC* $k\ s$) and state expectation (*ExpI* $k\ s$) are simply defined by recursion on the bounding value k. Soundness is proved by induction on the bounding value k and using the iterative equations given by Theorem 9.

Definition 22. *Computing pCTL-satisfiability (2)*

$$\begin{array}{ll}
\mathsf{ProbUb}\ 0\ s\ S_1\ S_2 & := \mathit{if}\ s \in S_2\ \mathit{then}\ 1\ \mathit{else}\ 0 \\
\mathsf{ProbUb}\ (k+1)\ s\ S_1\ S_2 & := \mathit{if}\ s \in S_1 \setminus S_2\ \mathit{then}\ \sum_{s' \in S} \tau\ s\ s' \cdot \mathsf{ProbUb}\ k\ s'\ S_1\ S_2 \\
 & \qquad\qquad\qquad\quad \mathit{else}\ (\mathit{if}\ s \in S_2\ \mathit{then}\ 1\ \mathit{else}\ 0)
\end{array}$$

$$\begin{array}{ll}
\mathsf{ExpC}\ 0\ s & := 0 \\
\mathsf{ExpC}\ (k+1)\ s & := \rho\ s + \sum_{s' \in S} \tau\ s\ s' \cdot (\iota\ s\ s' + \mathsf{ExpC}\ k\ s')
\end{array}$$

$$\begin{array}{ll}
\mathsf{ExpI}\ 0\ s & := \rho\ s \\
\mathsf{ExpI}\ (k+1)\ s & := \sum_{s' \in S} \tau\ s\ s' \cdot \mathsf{ExpI}\ k\ s'
\end{array}$$

$$\begin{array}{ll}
\mathsf{Sat}\ (P^{\bowtie r}\ (\Phi\ U^{\leq k}\ \Psi)) & := \mathit{do} \\
 & \quad F_1 \leftarrow \mathsf{Sat}\ \Phi \\
 & \quad F_2 \leftarrow \mathsf{Sat}\ \Psi \\
 & \quad \mathit{return}\ \{s \in S \mid \mathsf{ProbUb}\ k\ s\ F_1\ F_2 \bowtie r\} \\
\mathsf{Sat}\ (E^{\bowtie r}\ (C^{<k})) & := \mathit{return}\ \{s \in S \mid \mathsf{ExpC}\ k\ s \bowtie r\} \\
\mathsf{Sat}\ (E^{\bowtie r}\ (I^{=k})) & := \mathit{return}\ \{s \in S \mid \mathsf{ExpI}\ k\ s \bowtie r\}
\end{array}$$

Our next step is to check the unbounded until operator. Here we compute the probability $P_{\Phi,\Psi}(s) := \mu_s\{\omega \in \Omega \mid [\![\Phi\ U^\infty\ \Psi, s{\cdot}\omega]\!]_P\}$ for each state s by setting up a system of linear equations. From Theorem 9 and the behavior of the unbounded until operator we derive a system of linear equations for $P_{\Phi,\Psi}(s)$.

$$P_{\Phi,\Psi}(s) = \begin{cases} \sum_{s' \in E(s)} \tau\ s\ s' \cdot P_{\Phi,\Psi}(s') & \text{if} \quad s \in \Phi \setminus \Psi \\ 1 & \text{if} \quad s \in \Psi \\ 0 & \text{otherwise} \end{cases}$$

We show that such a linear equation system has a unique solution, with two conditions: (1) the solutions are equal on Ψ and (2) the solutions are equal in all states which never reach Ψ, i.e. $P_{\Phi,\Psi}(s) = 0$. We proved this lemma following the uniqueness proof in [6].

Lemma 23. *Unique solution*

$$\begin{array}{rl}
 & \Phi \subseteq S \wedge \Psi \subseteq N \subseteq S \\
\wedge & \forall s \in S.\ P_{\Phi,\Psi}(s) = 0 \longrightarrow s \in N \\
\wedge & \forall s \in S \setminus N.\ l_1\ s - c\ s = \sum_{s' \in S} \tau\ s\ s' \cdot l_1\ s' \\
\wedge & \forall s \in S \setminus N.\ l_2\ s - c\ s = \sum_{s' \in S} \tau\ s\ s' \cdot l_2\ s' \\
\wedge & \forall s \in N.\ l_1\ s = l_2\ s \\
\longrightarrow & \forall s \in S.\ l_1\ s = l_2\ s
\end{array}$$

To find a solution of such a system of linear equations, we formalized Gauss-Jordan elimination on matrices represented as functions [19]. Then we adapted this to use states as indices instead of natural numbers. Correctness says that if *gauss-jordan* $M\ a$ returns *Some* x, then x is a solution to the equation system $M \cdot x = a$.

Lemma 24. *Gauss-Jordan elimination*

$$\mathsf{gauss\text{-}jordan}\ M\ a = \mathsf{Some}\ x \longrightarrow \forall s \in S.\ \left(\sum_{s' \in S} M\ s\ s' \cdot x\ s'\right) = a\ s$$

Before we use the uniqueness of our system of linear equationss, Lemma 23 requires us to compute the states with $P_{\Phi,\Psi}(s) = 0$ before the algorithm builds the system of linear equations. $\mathsf{Prob0}$ computes the set of all states with $P_{\Phi,\Psi}(s) > 0$ and returns the complement. The set of all s with $P_{\Phi,\Psi}(s) > 0$ is computed by starting with $R = \Psi$ and adding states to R which are in Φ and are predecessors of a state in R. With Lemma 11 we know that R contains all reachable states, hence $P_{\Phi,\Psi}(s) > 0$ for all $s \in R$. The termination measure for the *while*-combinator is the difference $S \setminus R$, with each step either states are added, or the loop terminates.

Definition 25. *Compute* $[\![P^{=0}(\Phi\ U^{\infty}\ \Psi)]\!]$

$$\begin{aligned} \mathsf{pred}\ \Phi\ R\ &:= \{s \in \Phi \mid R \cap E(s) \neq \emptyset\} \\ \mathsf{Prob0}\ \Phi\ \Psi &:= S\ \setminus\ \mathit{while}\ (\lambda R.\ \neg \mathsf{pred}\ \Phi\ R \subseteq R)\ (\lambda R.\ R \cup \mathsf{pred}\ \Phi\ R)\ \Psi \end{aligned}$$

The system of linear equations solved by $\mathsf{gauss\text{-}jordan}\ M\ a$ needs to be in the right form, i.e. the matrix M contains all variable coefficients and a all constants. We introduce $\mathsf{LES}\ F$ to define the matrix of the linear equation system $l\ s = (\sum_{s \in S} \tau\ s\ s' \cdot l\ s') + a\ s$ for $s \notin F$, and $l\ s = a\ s$ if $s \in F$.

Definition 26. *Linear Equation System to Compute Unbounded Until*

$$\begin{aligned} \mathsf{LES}\ F\ r\ c\ := \mathit{if}\ r \in F\ &\mathit{then}\ (\mathit{if}\ c = r\ \mathit{then}\ 1\ \mathit{else}\ 0) \\ &\mathit{else}\ (\mathit{if}\ c = r\ \mathit{then}\ \tau\ r\ c - 1\ \mathit{else}\ \tau\ r\ c) \end{aligned}$$

Combining all this we can finally compute the probability of a unbounded until formula. We prove its soundness using Lemmas 24 and 23, and Theorem 9.

Definition 27. *Computing pCTL-satisfiability (3)*

$$\begin{aligned} \mathsf{Sat}\ (P^{\bowtie r}\ (\Phi\ U^{\infty}\ \Psi)) := \mathit{do}\ & \\ & F_1 \leftarrow \mathsf{Sat}\ \Phi \\ & F_2 \leftarrow \mathsf{Sat}\ \Psi \\ & p \leftarrow \mathsf{gauss\text{-}jordan}\ (\mathsf{LES}\ (F_2 \cup \mathsf{Prob0}\ F_1\ F_2)) \\ & \qquad\qquad (\lambda s.\ \mathit{if}\ s \in F_2\ \mathit{then}\ 1\ \mathit{else}\ 0) \\ & \mathit{return}\ \left\{s \in S \mid p\ s \bowtie r\right\} \end{aligned}$$

The last equation of Sat computes the unbounded reward $E^{\bowtie r}(F^{\infty}\ \Phi)$. Similar to the unbounded until operator, we introduce a system of linear equations for $R_{\Phi}(s) := \int_{\omega} [\![F^{\infty}\ \Phi, s{\cdot}\omega]\!]_E d\mu_s$. With Theorem 17 we know that $R_{\Phi}(s)$ is finite if $P_{S,\Phi}(s) = 1$. If $P_{S,\Phi}(s) < 1$ there is a non-zero probability that Φ is never reached, and hence $R_{\Phi}(s) = \infty$.

$$R_{\Phi}(s) = \begin{cases} \sum_{s' \in E(s)} \tau\ s\ s' \cdot (\rho\ s + \iota\ s\ s' + R_{\Phi}(s')) & \text{if} \quad P_{S,\Phi}(s) = 1 \wedge s \notin \Phi \\ 0 & \text{if} \quad s \in \Phi \\ \infty & \text{otherwise} \end{cases}$$

To be usable with *LES*, we rewrite the first equation into:

$$R_{\Phi}(s) - \left(\rho\ s + \sum_{s' \in E(s)} \tau\ s\ s' \cdot \iota\ s\ s'\right) = \sum_{s' \in E(s)} \tau\ s\ s' \cdot R_{\Phi}(s')\ .$$

The Gauss-Jordan elimination we use works only on real numbers, luckily we can replace ∞ by 0 and replace it again after we solved the equation system. This is sound since for each s and $s' \in E(s)$ with $R_{\Phi}(s') = \infty$ either $s \in \Phi$ or $R_{\Phi}(s) = \infty$ hold. The states s with $P_{S,\Phi}(s) = 1$ are computed by *Prob1*, building on *Prob0*.

Definition 28. *Compute* $[\![P^{=1}(\Phi\ U^{\infty}\ \Psi)]\!]$

$$\mathsf{Prob1}\ \Phi\ \Psi := \mathsf{Prob0}\ (\Phi \setminus \Psi)\ (\mathsf{Prob0}\ \Phi\ \Psi)$$

We know that the resulting states only reach states which again reach Ψ, hence the assumptions of Theorem 15 are fulfilled, and we know that *Prob1* $S\ \Phi$ is the set of all states s with $P_{S,\Phi}(s) = 1$. With all this, we can formalize the last equation for *Sat*.

Definition 29. *Computing pCTL-satisfiability (4)*

```
$\mathsf{Sat}\ (E^{\bowtie r}\ (F^{\infty}\ \Phi)) :=$ $do$
        $F \leftarrow \mathsf{Sat}\ \Phi$
        $let\ Y = \mathsf{Prob1}\ S\ F$
        $l \leftarrow \mathsf{gauss\text{-}jordan}\ (\mathsf{LES}\ (S \setminus (Y \setminus F)))$
            $(\lambda s.\ if\ i \in Y \setminus F\ then\ -(\rho\ s + (\sum_{s' \in S} \cdot\ \tau\ s\ s' \cdot \iota\ s\ s'))$
                                   $else\ 0)$
        $let\ e = (\lambda s.\ if\ s \in Y\ then\ l\ s\ else\ \infty)$
        $return\ \{s \in S \mid e\ s \bowtie r\}$
```

Finally we show the soundness of *Sat* by induction on the structure of Φ. If we assume that *Sat* terminates with a result F, then F is the same set as defined by the semantic.

Theorem 30. *Soundness of Sat*

$$\mathsf{Sat}\ \Phi = \mathsf{Some}\ F \longrightarrow [\![\Phi]\!] = F$$

Now we turn to completeness. The only case in which *Sat* returns *None* is when the Gauss-Jordan elimination does not find a unique solution. Hence we need the property that if a unique solution exists, then *gauss-jordan* returns this solution.

Theorem 31. *Completeness of gauss-jordan*
If there is a unique solution x for $M \cdot x = a$:

$$\forall s \in S.\ \sum_{s' \in S} M\ s\ s' \cdot x\ s' = a\ s$$

$$\forall y.\ \left(\forall s \in S.\ \sum_{s' \in S} M\ s\ s' \cdot y\ s' = a\ s\right) \longrightarrow \forall s \in S.\ x\ s = y\ s$$

then gauss-jordan *returns a result:*

$$\exists x'.\ \textsf{gauss-jordan}\ M\ a = \textsf{Some}\ x'$$

With this and Lemma 23 we prove that *Sat* always returns a result:

Theorem 32. *Completeness of Sat*

$$\exists F.\ \textsf{Sat}\ \Phi = \textsf{Some}\ F\ .$$

Using Theorem 30 we finally show

Corollary 33. *Soundness and completeness of Sat*

$$\textsf{Sat}\ \Phi = \textsf{Some}\ [\![\Phi]\!]\ .$$

5 Discussion

We used the tutorial [12] as a guideline to formalize the pCTL model checking algorithm. Most parts of the soundness proof are straightforward. Three parts, however, required a more substantial formalization of the background theory:

- The correctness of *Prob1* is based on Theorem 15, which required us to formalize state fairness as found in [1].
- For the unbounded until and the unbounded rewards we solve a linear equation system. We needed to show that the solution of this equation system is unique, for which we followed the original proof from [6].
- The unbounded reward for a state can only be characterized as a linear equation if the reward is finite. We needed Theorem 17 to show that the reward is finite, if the final states are almost always reached.

Technically, the largest difference between our work and Kwiatkowska *et. al.* [12] is the construction of the probability space of paths: we use infinite products of probability spaces, whereas they use Caratheodory on semi-rings of sets. We do not need to show that the probability of cylinders is countably additive, this is generically done for infinite products. We want to reuse the infinite products for continuous-time Markov chains and Markov decision processes. With Caratheodory on semi-rings of sets it would be necessary to show countably additivity for each of them. Nevertheless, we intend to formalize the latter construction, too, as it is a valuable addition to our library.

The equations we give for the algorithm are not directly executable by the code generator in Isabelle [5]. We use sets in our equations, and the adaption of Gauss-Jordan elimination uses an arbitrary mapping from $\{0, \ldots, |S| - 1\}$ to S. One method to obtain a executable version is to create a copy $\textsf{Sat}_L$ of *Sat* operating on lists instead of subsets of S. We assume as input a list of states $xs := [s_0, s_1, \ldots s_n]$, and define the Markov chains on $S :=$ set-of xs. It should be straightforward to show that $\textsf{Sat}\ \Phi = \textsf{Some}\ F$ implies $\textsf{set-of}\ (\textsf{Sat}_L\ \Phi) = F$. The biggest hurdle is the while-combinator in *Prob0* and the adaption of Gauss-Jordan elimination.

6 Conclusion

The formalization of pCTL model checking in a proof assistant opens up a number of possible application scenarios:

Model Checking as an Isabelle Proof Method. Once we have made our pCTL model checker executable as explained in Section 5, we can call it as an automated proof method for pCTL formulas within Isabelle. Of course this is only practical for small examples, for larger ones an external pCTL model checker would be used as an oracle that must be trusted.

Certified Model Checking. Result checking is an established technique where, rather than verifying an algorithm, each execution of the algorithm is checked. This requires the algorithm to return a checkable certificate. A particularly successful example of such a system architecture is CeTA [24], a checker for termination proofs which regularly finds bugs in termination proof tools. CeTA is verified in Isabelle and an efficient Haskell program is extracted that can check large proof certificates.

Verification of Parametrized Models. The Markov chain may depend on parameters like the number of parallel processes. Such parameterized models can be model checked only for fixed parameter values. Our theory allows one to formalize and verify such parameterized models for all possible parameter values interactively. As case studies we formalized IPv4 address allocation in the ZeroConf protocol and anonymity of the Crowds protocol [9]. The formalizations we describe in Section 3 where essential for these case studies.

The formalization is available in the AFP [9,19]. It has about 4480 lines: 3670 lines for the formalization of DTMCs, 270 lines for Gauss-Jordan elimination, and 1140 lines for pCTL model checking.

Our future goal is to formalize more probabilistic models with the corresponding model checking algorithms, like pCTL for Markov decision processes, continuous stochastic logic for continuous-time Markov chains and probabilistic timed CTL for probabilistic timed automata.

References

1. Baier, C.: On the Algorithmic Verification of Probabilistic Systems. Habilitation, Universität Mannheim (1998)
2. Bauer, H.: Probability Theory. de Gruyter (1995)
3. Chou, C.T., Peled, D.: Formal verification of a partial-order reduction technique for model checking. Journal of Automated Reasoning 23(3-4), 265–298 (1999)
4. Coble, A.R.: Anonymity, Information, and Machine-Assisted Proof. Ph.D. thesis, King's College, University of Cambridge (2009)
5. Haftmann, F., Nipkow, T.: Code Generation via Higher-Order Rewrite Systems. In: Blume, M., Kobayashi, N., Vidal, G. (eds.) FLOPS 2010. LNCS, vol. 6009, pp. 103–117. Springer, Heidelberg (2010)
6. Hansson, H., Jonsson, B.: A logic for reasoning about time and reliability. Tech. Rep. SICS/R90013, Swedish Institute of Computer Science (December 1994)

7. Hurd, J.: Formal Verification of Probabilistic Algorithms. Ph.D. thesis, University of Cambridge (2002)
8. Hölzl, J., Heller, A.: Three Chapters of Measure Theory in Isabelle/HOL. In: van Eekelen, M.C.J.D., Geuvers, H., Schmaltz, J., Wiedijk, F. (eds.) ITP 2011. LNCS, vol. 6898, pp. 135–151. Springer, Heidelberg (2011)
9. Hözl, J., Nipkow, T.: Markov models. In: Klein, G., Nipkow, T., Paulson, L. (eds.) The Archive of Formal Proofs, formal proof development (January 2012), `http://afp.sf.net/entries/Markov_Models.shtml`
10. Katoen, J.P., Zapreev, I.S., Hahn, E.M., Hermanns, H., Jansen, D.N.: The ins and outs of the probabilistic model checker MRMC. Performance Evaluation 68, 90–104 (2011)
11. Klein, G., Elphinstone, K., Heiser, G., Andronick, J., Cock, D., Derrin, P., Elkaduwe, D., Engelhardt, K., Kolanski, R., Norrish, M., Sewell, T., Tuch, H., Winwood, S.: seL4: Formal verification of an OS kernel. In: Proc. 22nd ACM Symposium on Operating Systems Principles 2009, pp. 207–220 (2009)
12. Kwiatkowska, M., Norman, G., Parker, D.: Stochastic Model Checking. In: Bernardo, M., Hillston, J. (eds.) SFM 2007. LNCS, vol. 4486, pp. 220–270. Springer, Heidelberg (2007)
13. Kwiatkowska, M., Norman, G., Parker, D.: PRISM 4.0: Verification of Probabilistic Real-Time Systems. In: Gopalakrishnan, G., Qadeer, S. (eds.) CAV 2011. LNCS, vol. 6806, pp. 585–591. Springer, Heidelberg (2011)
14. Lammich, P., Müller-Olm, M., Wenner, A.: Predecessor Sets of Dynamic Pushdown Networks with Tree-Regular Constraints. In: Bouajjani, A., Maler, O. (eds.) CAV 2009. LNCS, vol. 5643, pp. 525–539. Springer, Heidelberg (2009)
15. Leroy, X.: A formally verified compiler back-end. J. Automated Reasoning 43, 363–446 (2009)
16. Levin, D.A., Peres, Y., Wilmer, E.L.: Markov chains and mixing times. AMS (2006)
17. Liu, L., Hasan, O., Tahar, S.: Formalization of Finite-State Discrete-Time Markov Chains in HOL. In: Bultan, T., Hsiung, P.-A. (eds.) ATVA 2011. LNCS, vol. 6996, pp. 90–104. Springer, Heidelberg (2011)
18. Mhamdi, T., Hasan, O., Tahar, S.: Formalization of Entropy Measures in HOL. In: van Eekelen, M.C.J.D., Geuvers, H., Schmaltz, J., Wiedijk, F. (eds.) ITP 2011. LNCS, vol. 6898, pp. 233–248. Springer, Heidelberg (2011)
19. Nipkow, T.: Gauss-Jordan elimination for matrices represented as functions. In: Klein, G., Nipkow, T., Paulson, L. (eds.) The Archive of Formal Proofs, formal proof development (August 2011), `http://afp.sf.net/entries/Gauss-Jordan-Elim-Fun.shtml`
20. Nipkow, T., Paulson, L.C., Wenzel, M.: Isabelle/HOL. LNCS, vol. 2283. Springer, Heidelberg (2002)
21. Reif, W., Schellhorn, G., Vollmer, T., Ruf, J.: Correctness of efficient real-time model checking. J. UCS 7(2), 194–209 (2001)
22. Schimpf, A., Merz, S., Smaus, J.-G.: Construction of Büchi Automata for LTL Model Checking Verified in Isabelle/HOL. In: Berghofer, S., Nipkow, T., Urban, C., Wenzel, M. (eds.) TPHOLs 2009. LNCS, vol. 5674, pp. 424–439. Springer, Heidelberg (2009)
23. Sprenger, C.: A Verified Model Checker for the Modal μ-Calculus in Coq. In: Steffen, B. (ed.) TACAS 1998. LNCS, vol. 1384, pp. 167–183. Springer, Heidelberg (1998)
24. Thiemann, R., Sternagel, C.: Certification of Termination Proofs Using CeTA. In: Berghofer, S., Nipkow, T., Urban, C., Wenzel, M. (eds.) TPHOLs 2009. LNCS, vol. 5674, pp. 452–468. Springer, Heidelberg (2009)