Parametric Verification of Weighted Systems

April 11, 2015

Peter Christoffersen, **Mikkel Hansen**, Anders Mariegaard, Julian T. Ringsmose, Kim G. Larsen & Radu Mardare

Department of Computer Science
Aalborg University
Denmark

AALBORG UNIVERSITY
DENMARK

Motivation

In recent time a lot of effort have been put into capturing important characteristics of real world systems in various modeling formalisms:

- Time: timed automata, timed CCS.
- Uncertainty: probabilistic systems based on markov chains.
- Costs and resources: weighted systems.

Each modeling formalism is associated with specification languages to verify requirements.

Motivation

Another important characteristic, usually neglected, is that of *incomplete information*:

- Models containing guesses and estimations for time usage, cost and probabilistic behavior.
- Difficult verification of properties as it depend on the inaccurate model.

Related Work

Our work is inspired by:

- Parametric Kripke Structure and Parametric CTL by Sathawornwichit et al.
- Symbolic Dependency Graphs by Jensen et al.

Contribution

Based on parametric extensions to weighted transition systems and weighted CTL we:

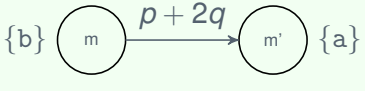
- Define Parametric Dependency Graphs (PDGs) to structurally represent dependencies.
- Show how to compute minimal fixed points of *assignments* to nodes in a PDG in a finite number of steps.
- Prove that computing the fixed points solves the model checking problem.

Finally, we have implemented a web-based prototype tool deriving parameter constraints (`pvtool.dk`).

Parameters

We allow linear expressions over parameters to be used as transition weights and as upper bounds on path formulae.

Model



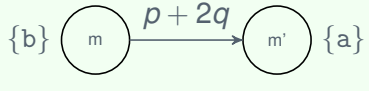
Formula

$A \top U_{\leq p}$ a

Parameters

We allow linear expressions over parameters to be used as transition weights and as upper bounds on path formulae.

Model



Formula

$A \top U_{\leq p}$ a

This changes the model checking problem into a problem of finding good parameter values.

Model checking

m satisfies $A \top U_{\leq p}$ a
if $p + 2q \leq p$.

Interpretations

We assume a finite set of parameters $\mathcal{P}$ and linear expressions in parameters.

Interpretations

$$\boldsymbol{i} : \mathcal{P} \longrightarrow \mathbb{N}$$

Interpretations are extended to linear expressions in parameters i.e if $\boldsymbol{i}(p) = 5$ and $\boldsymbol{i}(q) = 7$ then

$$\boldsymbol{i}(2p + 5q + 1) = 46$$

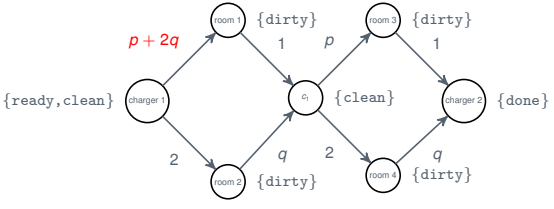
The set of all linear expressions is denoted by $\mathcal{E}$.

Parametric Weighted Transition System

Definition

A Parametric Weighted Transition System (PTS) $\mathcal{M}$ is a triple $\mathcal{M} = (M, \rightarrow, \ell)$, where

- M is a finite non-empty set of *states*.
- $\rightarrow \subseteq M \times \mathcal{E} \times M$ is the transition relation.
- $\ell : M \longrightarrow 2^{\mathcal{AP}}$ is a labeling function mapping states in M to a set of atomic propositions



Parametric Weighted CTL

We extend weighted CTL with upper bound parametric constraints:

Definition

The set of PTL *state formulae* are given by the abstract syntax:

$$\Phi, \Psi ::= \top \mid \bot \mid \mathtt{a} \mid \Phi \wedge \Psi \mid \Phi \vee \Psi \mid E\varphi \mid A\varphi$$

and the set of PTL *path formulae* are given by the abstract syntax:

$$\varphi ::= X_{\leq e}\Phi \mid \Phi U_{\leq e}\Psi$$

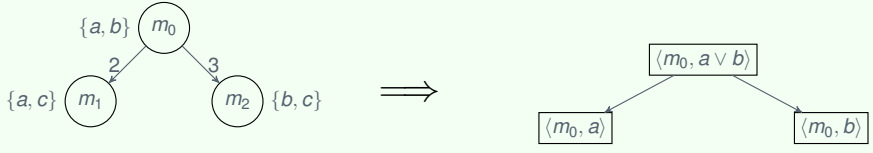
where $\mathtt{a} \in \mathcal{AP}$ and $e \in \mathcal{E}$.

$$\rho = \underbrace{m_0 \overset{\Phi}{e_1} m_1 \overset{\Phi}{\cdots} e_j \overset{\Psi}{m_j} \cdots}_{\leq e}$$

Dependency Graphs
Intuition

Dependency graphs are traditionally used to encode dependencies between properties, where the structure of the graph can be seen as a graphical representation of a formula's semantics.

Example



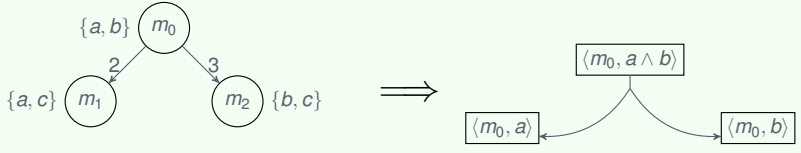
$\mathcal{M}, m_0 \models a \vee b$

Either disjunct must be satisfied, represented by two hyper-edges: one for each disjunct.

Dependency Graphs
Intuition

Dependency graphs are traditionally used to encode dependencies between properties, where the structure of the graph can be seen as a graphical representation of a formula's semantics.

Example



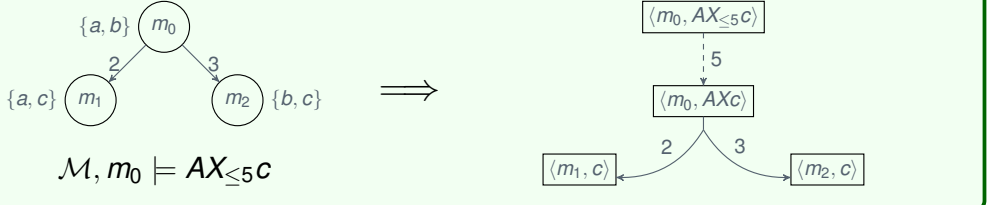
$\mathcal{M}, m_0 \models a \wedge b$

Both conjuncts must be satisfied, represented by a single hyper-edge going to both conjuncts.

Symbolic Dependency Graphs
Intuition

Symbolic dependency graphs can be used as an abstraction of problems of problems with quantitative dependencies.

Example



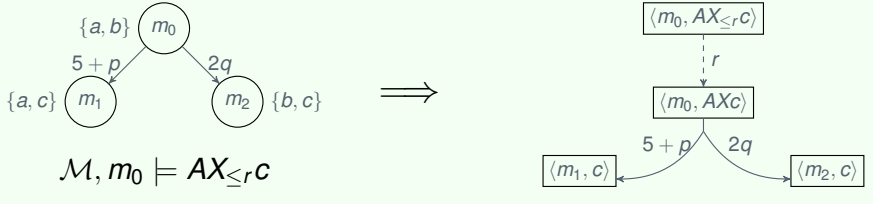
The bound is abstracted away by the cover-edge.

The minimal cost of satisfying the (unbounded) formula is accumulated at the node below the cover-edge and checked against the cover-edge weight.

Parametric Dependency Graphs
Intuition

We propose a parametric extension to symbolic dependency graphs called parametric dependency graphs that allows linear expressions involving parameters as weights.

Example



Besides allowing linear expressions involving parameters, parametric dependency graphs works just like symbolic dependency graphs.

Parametric Dependency Graphs
Definition

Definition

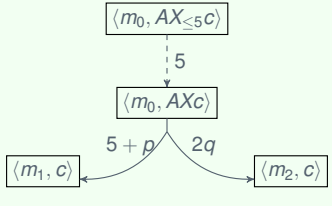
A Parametric Dependency Graph (PDG) is a tuple $\mathcal{G} = (N, H, C)$, where

- N is a finite set of nodes,
- $H \subseteq N \times 2^{\mathcal{E} \times N}$ is a finite set of hyper-edges and
- $C \subseteq N \times \mathcal{E} \times N$ is a finite set of cover-edges

Whenever $(n, T) \in H$ we refer to n as the source node and T as the target-set.

For each $n' \in T$ we refer to n' as a target-node, or simply target.

Example



Assignments
Definition

We use assignments to encode the parametric cost of reaching a truth value in the PDG.

Definition

Given a PDG $\mathcal{G} = (N, H, C)$, an assignment

$$A : N \longrightarrow (\mathfrak{I} \longrightarrow \mathbb{N} \cup \{\infty\})$$

on $\mathcal{G}$ is a mapping from each node $n \in N$ to a function that, given a parameter interpretation, yields a natural number or ∞.

We denote the set of all assignments $\mathfrak{A}$.

We use 0 to represent "good" values, i.e. satisfiable, and ∞ to represent "bad" values, i.e. non-satisfiable.

Assignments
Ordering

Definition

$(\mathfrak{A}, \sqsubseteq)$ is a poset such that for $A_1, A_2 \in \mathfrak{A}$:

$$A_1 \sqsubseteq A_2 \quad \text{iff} \quad \forall n \in N \, \forall \boldsymbol{i} \in \mathfrak{I} : A_1(n)(\boldsymbol{i}) \geq A_2(n)(\boldsymbol{i})$$

A^∞ denotes the assignment that maps to node a function that assigns the value ∞ regardless of parameter interpretations, i.e.

$$\forall n \in N \, \forall \boldsymbol{i} \in \mathfrak{I} : A^\infty(n)(\boldsymbol{i}) = \infty$$

Assignments
Ordering

Definition

$(\mathfrak{A}, \sqsubseteq)$ is a poset such that for $A_1, A_2 \in \mathfrak{A}$:

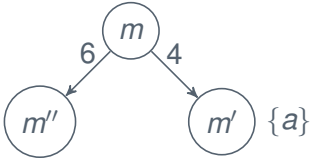
$$A_1 \sqsubseteq A_2 \quad \text{iff} \quad \forall n \in N \, \forall \boldsymbol{i} \in \mathfrak{I} : A_1(n)(\boldsymbol{i}) \geq A_2(n)(\boldsymbol{i})$$

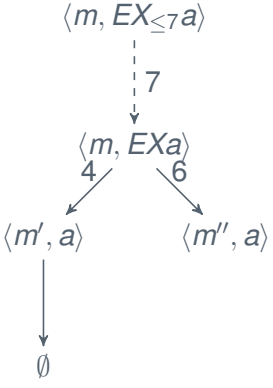
A^∞ denotes the assignment that maps to node a function that assigns the value ∞ regardless of parameter interpretations, i.e.

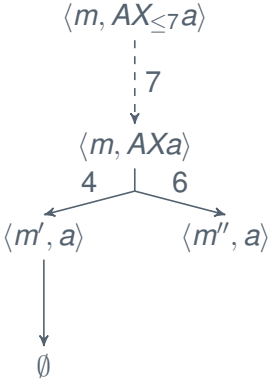
$$\forall n \in N \, \forall \boldsymbol{i} \in \mathfrak{I} : A^\infty(n)(\boldsymbol{i}) = \infty$$

$(\mathfrak{A}, \sqsubseteq)$ constitutes a complete lattice with A^∞ as bottom element.

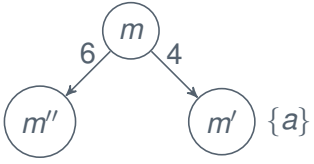
Understanding Assignments

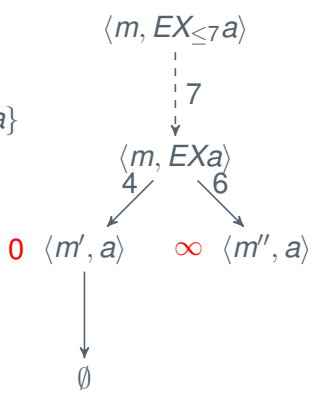


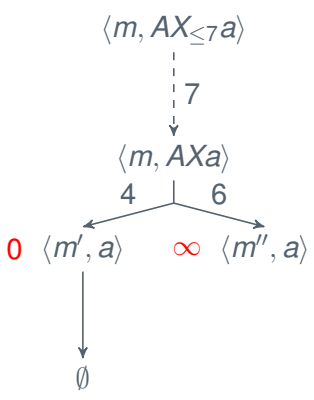




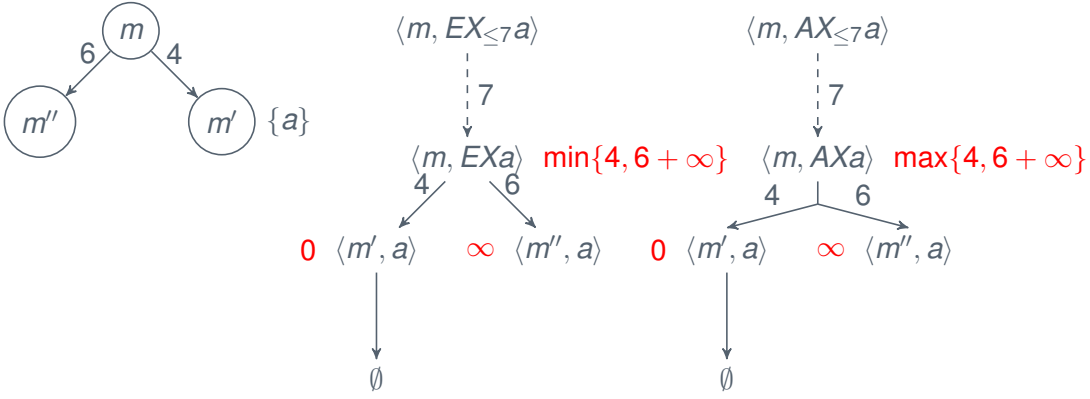
Understanding Assignments



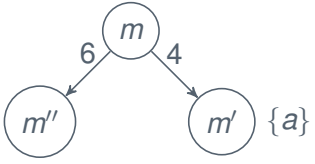


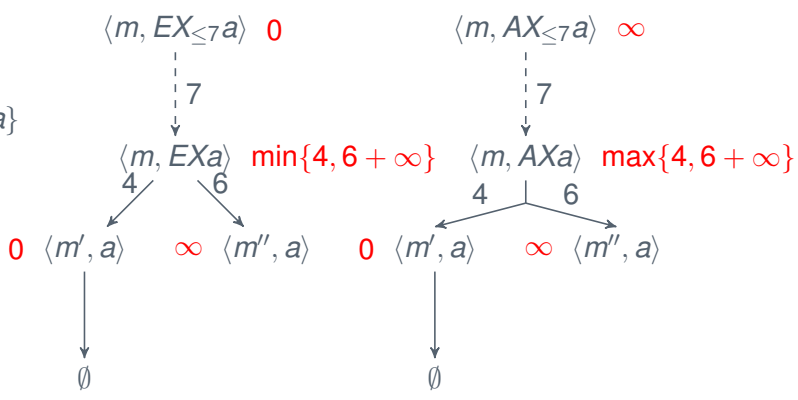


Understanding Assignments



Understanding Assignments





Model Checking
Updating assignments

A global update function iteratively updates the PDG node assignments.
Let $\min\{\emptyset\} = \infty$ and $\max\{\emptyset\} = 0$.

Definition

Given a PDG $\mathcal{G} = (N, H, C)$, $\boldsymbol{F} : \mathfrak{A} \longrightarrow \mathfrak{A}$ is a function that given an assignment on $\mathcal{G}$ produces a new assignment on $\mathcal{G}$, defined as follows:

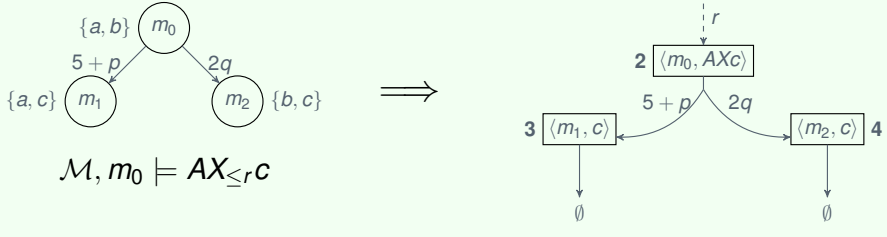
$$\boldsymbol{F}(A)(n)(\boldsymbol{i}) = \begin{cases} \begin{cases} 0 & \text{if } A(n')(\boldsymbol{i}) \leq \boldsymbol{i}(e) \\ \infty & \text{otherwise} \end{cases} & \text{if } n \overset{e}{\dashrightarrow} n' \\ \min\limits_{(n,T)\in H} \{ \max\limits_{(e,n')\in T} \{A(n')(\boldsymbol{i}) + \boldsymbol{i}(e)\}\} & \text{otherwise} \end{cases}$$

We let $\boldsymbol{F}^i(A)$ denote i repeated applications of $\boldsymbol{F}$ on A.

Model Checking
Example

We encode the model checking problem, $\mathcal{M}, m_0 \models AX_{\leq r} c$ in a PDG.

Example

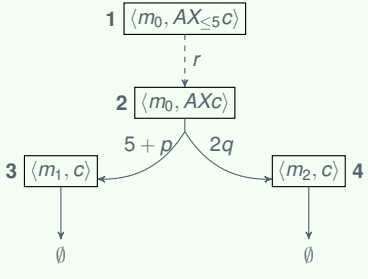


Model Checking
Example

By iteratively applying $\boldsymbol{F}$ on A^{∞} we compute the fixed point assignment A^{min}.

Example

n	1	2	3	4
A^{∞}	∞	∞	∞	∞

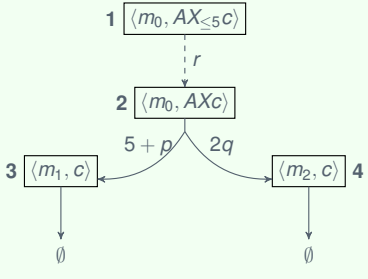


Model Checking
Example

By iteratively applying $\boldsymbol{F}$ on A^{∞} we compute the fixed point assignment A^{min}.

Example

n	1	2	3	4
A^{∞}	∞	∞	∞	∞
$\boldsymbol{F}(A^{\infty})$	$\begin{cases} 0 & \text{if } \infty \leq r \\ \infty & \text{otherwise} \end{cases}$	∞	0	0



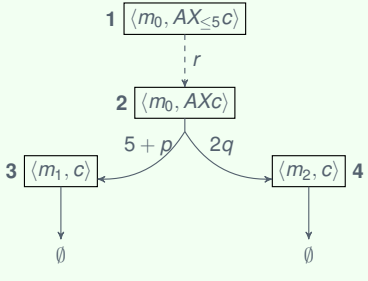
In the first iteration, only nodes with a hyper edge going to an empty target-set gets updated (nodes 3 and 4).

Model Checking
Example

By iteratively applying $\boldsymbol{F}$ on A^{∞} we compute the fixed point assignment A^{min}.

Example

n	1	2	3	4
A^{∞}	∞	∞	∞	∞
$\boldsymbol{F}(A^{\infty})$	$\begin{cases} 0 & \text{if } \infty \leq r \\ \infty & \text{otherwise} \end{cases}$	∞	0	0
$\boldsymbol{F}^2(A^{\infty})$	$\begin{cases} 0 & \text{if } \infty \leq r \\ \infty & \text{otherwise} \end{cases}$	$\max\{5+p, 2q\}$	0	0



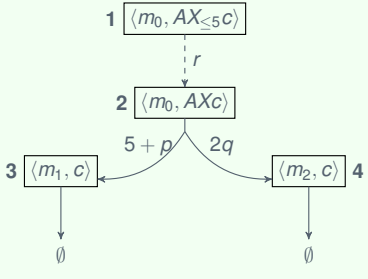
In the next iteration, node 2 is updated as it depends on nodes 3 and 4.

Model Checking
Example

By iteratively applying $\boldsymbol{F}$ on A^{∞} we compute the fixed point assignment A^{min}.

Example

n	1	2	3	4
A^{∞}	∞	∞	∞	∞
$\boldsymbol{F}(A^{\infty})$	$\begin{cases} 0 & \text{if } \infty \leq r \\ \infty & \text{otherwise} \end{cases}$	∞	0	0
$\boldsymbol{F}^2(A^{\infty})$	$\begin{cases} 0 & \text{if } \infty \leq r \\ \infty & \text{otherwise} \end{cases}$	$\max\{5+p, 2q\}$	0	0
$\boldsymbol{F}^3(A^{\infty})$	$\begin{cases} 0 & \text{if } \max\{5+p, 2q\} \leq r \\ \infty & \text{otherwise} \end{cases}$	$\max\{5+p, 2q\}$	0	0



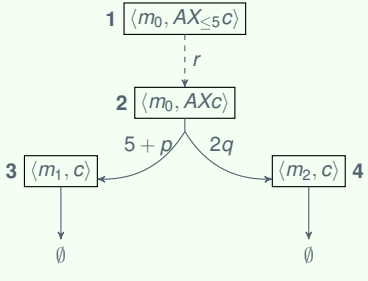
Now we have a weight to compare against the bound on the cover-edge, so node 1 gets updated.

Model Checking
Example

By iteratively applying $\boldsymbol{F}$ on A^{∞} we compute the fixed point assignment A^{min}.

Example

n	1		2	3	4
A^{∞}	∞		∞	∞	∞
$\boldsymbol{F}(A^{\infty})$	$\begin{cases} 0 \\ \infty \end{cases}$	if $\infty \leq r$ otherwise	∞	0	0
$\boldsymbol{F}^2(A^{\infty})$	$\begin{cases} 0 \\ \infty \end{cases}$	if $\infty \leq r$ otherwise	$\max\{5+p, 2q\}$	0	0
$\boldsymbol{F}^3(A^{\infty})$	$\begin{cases} 0 \\ \infty \end{cases}$	if $\max\{5+p, 2q\} \leq r$ otherwise	$\max\{5+p, 2q\}$	0	0
$\boldsymbol{F}^4(A^{\infty})$	$\begin{cases} 0 \\ \infty \end{cases}$	if $\max\{5+p, 2q\} \leq r$ otherwise	$\max\{5+p, 2q\}$	0	0



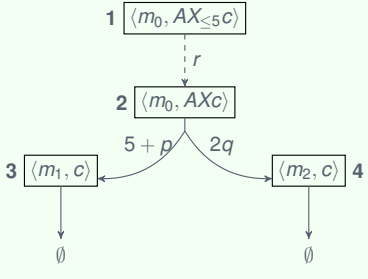
We see that the assignment doesn't change, and as such a fixed point has been reached.

Model Checking
Example

By iteratively applying $\boldsymbol{F}$ on A^{∞} we compute the fixed point assignment A^{min}.

Example

n	1		2	3	4
A^{∞}	∞		∞	∞	∞
$\boldsymbol{F}(A^{\infty})$	$\begin{cases} 0 \\ \infty \end{cases}$	$\begin{array}{l} \text{if } \infty \leq r \\ \text{otherwise} \end{array}$	∞	0	0
$\boldsymbol{F}^2(A^{\infty})$	$\begin{cases} 0 \\ \infty \end{cases}$	$\begin{array}{l} \text{if } \infty \leq r \\ \text{otherwise} \end{array}$	$\max\{5+p, 2q\}$	0	0
$\boldsymbol{F}^3(A^{\infty})$	$\begin{cases} 0 \\ \infty \end{cases}$	$\begin{array}{l} \text{if } \max\{5+p, 2q\} \leq r \\ \text{otherwise} \end{array}$	$\max\{5+p, 2q\}$	0	0
$\boldsymbol{F}^4(A^{\infty})$	$\begin{cases} 0 \\ \infty \end{cases}$	$\begin{array}{l} \text{if } \max\{5+p, 2q\} \leq r \\ \text{otherwise} \end{array}$	$\max\{5+p, 2q\}$	0	0



$$\max\{5+p, 2q\} \leq r \implies (5+p \leq r) \wedge (2q \leq r)$$

Model Checking
Fixed point assignments

Lemma

The update function $\boldsymbol{F}$ is monotone on the complete lattice $(\mathfrak{A}, \sqsubseteq)$.

According to Tarski's Fixed Point Theorem, $\boldsymbol{F}$ must have unique minimal fixed point which we denote A^{min}.

Theorem

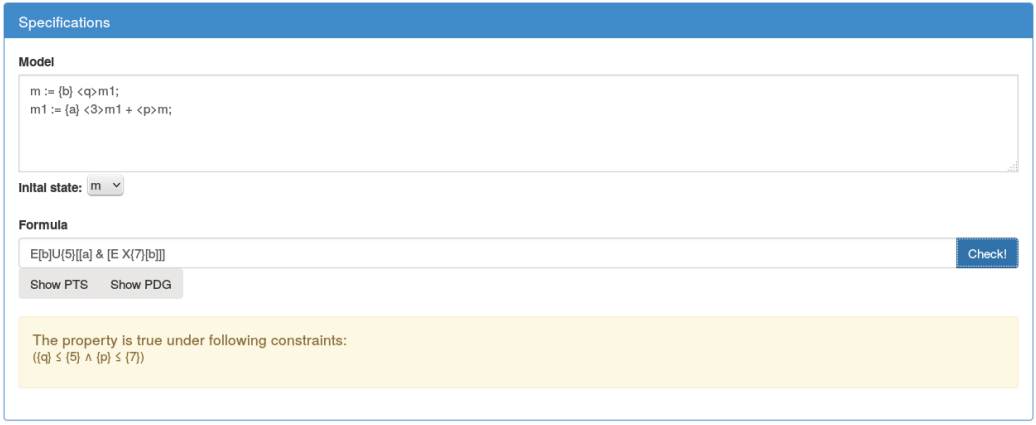
There exists a natural number i such that $A^{min} = \boldsymbol{F}^i(A^\infty)$.

Termination is guaranteed by the fact that assignments consitute a well-quasi -ordering and therefore cannot exist infinite antichains.

Prototype Tool

PVTool

As a proof of concept of the method for model checking of *PTL* on *PTS* presented in this work we have implemented a tool on `pvtool.dk`



Conclusion

We have developed an algorithm for computing minimal fixed points on PDGs and show that our technique can be used to:

- Solve model checking using parametric weighted CTL with upper-bound constraints.

Tool support and preliminary experiments were made to asses the feasibility of our approach.

Future Work

- Local algorithm instead of global.
- Other applications of our technique.
- Minimization of constraints computed by our tool.
- Proving (non)-existence of solution to parameter constraints.