Technische Universität Dresden • Fakultät Informatik

Data Preperation for PMC-Visualization

Bachelorarbeit zur Erlangung des ersten Hochschulgrades

Bachelor of Science (B.Sc.)

vorgelegt von

Franz Martin SCHMIDT

(geboren am 7. April 1999 in Halle (Saale))

Tag der Einreichung: June 17, 2023

Dipl. Inf Max Korn (Theoretische Informatik)

Contents

Abstract

Lorem ipsum

1 Introduction

PMC is short for Probabilistic Model Checking.

Model Checking is... Probabilistic means...

When analyzing PMCs a result table is generated which consists of result vectors. These result vectors for example contain probability values for temporal events, expected values of random variables and other qualitative and quantitative results. For central measure so called schedulers can be used to resolve non-determinism in the MDPs which in addition to the result provide one ore more optimal actions per state.

Already simple system models can lead to complex system behavior and an immense amount of states. An interactive visualization of an MDP can be used in support of understanding, analyzing and reviewing these complex systems. It can even be made use of to achieve further goals such as debugging or model repair. However, the pure data volume remains unaffected. The complexity is shifted to the visualization which has some general techniques to represent large amounts of data in a suitable way.

WHAT EXACTLY CAN BE DONE/HAS BEEN DONE —¿ INSERT HERE

In alternative to these general approaches domain specific preprocessing can be used to gain certain views on a given MDP that display areas or sets of states in summarized more compact form. For this purpose structural properties as well as criteria obtained from the result vectors and their containing optimal actions can be utilized.

This thesis is about the development, formalization and implementation of a collection of different *views*. Views are supposed to be applicable to only a subset of a given MDP and be composable with other views The input for a view is the result table of the MDP, whereas the output is a new collumn in the MDP result table that determines which states are to be grouped to a new one. The evaluation is performed within the existing web-based prototype of a PMC visualization platform (PMC-Viz) and on the basis of different MDP models. They are to be analyzed regarding the suitability to facilitate the understanding of complex operational models incl. analysis results and to support processes of debugging, model repair or strategy synthesis.

2 Preliminaries

2.1 Transition Systems

Motivation of transition systems

The following definition is directly taken form Principles of Modelchecking, Baier p. 20

Definition 2.1. A *transition system* TS is a tuple $(S, Act, \longrightarrow, I, AP, L)$ where

- S is a set of states,
- Act is a set of actions,
- $\longrightarrow \subseteq S \times Act \times S$ is transition relation,
- $I \subseteq S$ is a set of initial states,
- AP is a set of atomic propositions, and
- $L : S \to \mathcal{P}(AP)$

A transition system is called *finite* if S, AP and L are finite.

explanation of components

2.2 Markov Chain

NOTES BEGIN

- Markov Chain (MC)
- transition systems to markov chains: nondeterministic choices replaced by probablistic
- successor chosen according to probability distribution
- distribution only dependent on current state s (not path)
- system evolution not dependent on history but only current state → *memory-less property*

NOTES END

Definition 2.2. A *(discrete-time) Markov chain* is a tuple $\mathcal{M} = (S, \mathbf{P}, \iota_{init}, AP, L)$ where

- S is a countable, nonempty set of states,
- $\mathbf{P} : S \times S \to [0, 1]$ is the *transition probability function*, such that for all states s:

$$\sum_{s' \in S} \mathbf{P}(s, S') = 1.$$

- $\iota_{init} : S \to [0, 1]$ is the *initial distribution*, such that $\sum_{s \in S} \iota_{init}(s) = 1$, and
- AP is a set of atomic propositions and,
- $L : S \to \mathcal{P}(AP)$ a labeling function.

$\mathcal{M}$ is called *finite* if S and AP are finite. For finite $\mathcal{M}$, the *size* of $\mathcal{M}$, denoted $size(\mathcal{M})$, is the number of states plus the number of pairs $(s, s') \in S \times S$ with $\mathbf{P}(s, s') > 0$.

NOTES BEGIN

- Probability Function **P** specifies for each state s the probability **P** (s,s') of moving from s to s' in one step.
- constraint on **P** ensures that **P** is distribution
- ι_{init} (s) specifies system evolution starts in s
- states s with $\iota_{init}(s) > 0$ are considered *initial states*
- states s' with $\mathbf{P}(s, s') > 0$ are view as possible successors of s
- has no actions
  "As compositional approaches for Markov models are outside the scope of this monograph, actions are irrelevant in this chapter and are therefore omitted."

NOTES END

2.3 Markov Decision Process

NOTES BEGIN

- Markov decision process (MDP)
- idea: Adding nondeterminism to markov chains. MDPs permit both probabilistic and nondeterministic choices
- probabilistic choices: possible outcomes for of randomized actions -¿ requires statistical experiments to obtain adequate distributions that model average behavior of the environment
- information not available or guarantee about system properties is required -¿ nondeterminism
- Another example: randomized distributed algorithms. Non-determinism: interleaving behavior: nondeterministic choice which process, probabilistic: have rather restricted set of actions that have a random nature

- used for abstraction in markov chains: states grouped by AP and have a wide range of transition probabilities -¿ essentially nondeterminism -¿ transition probabilities are replaced by nondeterminism

NOTES END

Definition 2.3. A *Markov decision process* is a tuple $\mathcal{M} = (S, Act, \mathbf{P}, \iota_{init}, AP, L)$ where

- S is a countable set of states,
- Act is a set of actions,
- $\mathbf{P} : S \times Act \times S \to [0, 1]$ is the transition probability function such that for all states $s \in S$ and actions $\alpha \in Act$:

$$\sum_{s' \in S} \mathbf{P}(s, \alpha, s') \in \{0, 1\},$$

- $\iota_{init} : S \to [0, 1]$ is the initial distribution such that $\sum_{s \in S} \iota_{init}(s) = 1$,
- AP is a set of atomic propositions and
- $L : S \to \mathcal{P}(AP)$ a labeling function.

An action α is *enabled* in state s if and only if $\sum_{s' \in S} \mathbf{P}(s, \alpha, s') = 1$. Let $Act(s)$ denote the set of enabled actions in s. For any state $s \in S$, it is required that $Act(s) \neq \emptyset$. Each state s' for which $\mathbf{P}(s, \alpha, s') > 0$ is called an α-*successor* of s.

An MDP is called *finite* if S, Act and AP are finite.

NOTES BEGIN

- $\mathbf{P}(s, \alpha, t)$ can be arbitrary real numbers in $[0, 1]$ (sum up to 1 or 0 for fixed s and α), for algorithmic purposes rational
- unique initial distribution ι_{init}. Could be generalized to set of ι_{init} with nondeterministic choice at the beginning. For sake of simplicity: one single distribution
- operational behavior:
  - starting state s_0 yielded by ι_{init} with $\iota_{init}(s_0) > 0$
  - nondeterministic choice of enabled action (i.e. Probability sums up to one)
  - probabilistic choice of state (action fixed by nondeterministic selection)
- $MC = MDP \iff \forall s \in S : |Act(s)| = 1$
- $\implies$ MCs are a proper subset of MDPs

NOTES END

CONVENTION BEGIN

For $s_1, s_2 \in S$ and $\alpha \in Act$ we will write $(s_1, \alpha, s_2) \in \mathbf{P}$ if and only if $\mathbf{P}(s_1, \alpha, s_2) > 0$, that is to say if there is a non-zero probability of moving from state s_1 to state s_2 with action α. Analogously we will write $s \in \iota_{init}$ if and only if $\iota_{init}(s) > 0$ which means that there is chance of starting in that state.

not sure if I should: $I := \iota_{init}$ thereby meaning the underlying set

CONVENTION END TEST

3 View

Views are the central objective of this thesis. The purpose of a view is to obtain a simplification of a given an MDP. It is an independent an MDP derived from a given an MDP and represents a (simplified) view on the given one - hence the name. Thereby the original an MDP is retained.

In the preliminaries transition systems and Markov Chains were listed as simpler version of MDPs. More specifically transition systems and MDPs are special MDPs namely that have no probability distribution in each state for each action or no actions. When defining views it seems feasible to do so for the most general system of the ones of consideration.That is why we will define views on MDPs. Only for specific views and their implementation it is to be kept in mind that if they regard an action or the probability distribution of action in a state it is not applicable to transitions systems or MCs respectively. topic init states

3.1 Grouping Function

The conceptional idea of a view is to group states by some criteria and structure the rest of the system accordingly. To formalize the grouping we define a dedicated function.

Definition 3.1. Let $\mathcal{M} = (S, Act, \mathbf{P}, \iota_{init}, AP, L)$ be an MDP an M be and arbitrary set. We call any function $F_\theta : S \to M$ a *grouping function*. The symbol θ is a unique identifier.

The identifier θ normally hints the objective of the grouping function. Two states are grouped (should be Definition?) to a new state if and only if the grouping function maps them to the same value. The definition offers an easy way of defining groups of states. It is also very close to the actual implementation later on. The exact mapping depends on the desired grouping. In order to define a new set of states for the view, we define an equivalence relation R based on a given grouping function F_θ.

Definition 3.2. Let F_θ be a grouping function. We define the equivalence relation $R := \{(s_1, s_2) \in S \times S \mid F_\theta(s_1) = F_\theta(s_2)\}$

R is an equivalence relation because the equality relation is one. The property directly conveys to R. We observe that two states s_1, s_2 are grouped to a new state if and only if $(s_1, s_2) \in R$. This is the case if and only if $s_1, s_2 \in [s_1]_R = [s_2]_R$ where $[s_i]_R$ for $i \in \{1, 2\}$ denotes the equivalence class of R.

3.2 Formal Definition

The definition of a view is dependent on a given MDP and a grouping function F_θ. We derive the equivalence relation R as in Definition 3.2 and use its equivalence classes $[s]_R$ ($s \in S$) as states for the view. The rest of the an MDP is structured accordingly.

Definition 3.3. Let $\mathcal{M} = (S, Act, \mathbf{P}, \iota_{init}, AP, L)$ be an MDP and F_θ a grouping function. A *view* is an MDP $\mathcal{M}_\theta = (S', Act', \mathbf{P}', \iota_{init}{}', AP', L')$ that is derived from $\mathcal{M}$ with the grouping function F_θ where

- $S' = \{[s]_R \mid s \in S\}$
- $Act' = Act$
- $\mathbf{P}' = \{([s_1]_R, \alpha, [s_2]_R) \mid \exists s_1 \in [s_1]_R \exists s_2 \in [s_2]_R : ([s_1]_R, \alpha, [s_2]_R) \in \mathbf{P}\}$
- $\iota_{init}{}' = \{[s']_R \in S' \mid \exists s \in [s']_R : s \in \iota_{init}\}$
- $L' : S' \to \mathcal{P}(AP), [s]_R \mapsto \bigcup_{s \in [s]_R} \{L(s)\}$

and R is the equivalence relation according to Definition 3.2.

The identifier θ is inherited from F_θ to $\mathcal{M}_\theta$. It thereby uniquely determines $\mathcal{M}_\theta$ and declares that $\mathcal{M}_\theta$ is based on F_θ. Based on denotes that F_θ is the grouping function of $\mathcal{M}_\theta$ referred to in Definition 3.3.

Note that the definition is in a most general form in the sense that if in a view a property accounts to one piece of some entity the whole entity receives the property i.e.

- $(s_1, \alpha, s_2) \in \mathbf{P} \Rightarrow ([s_1]_R, \alpha, [s_2]_R) \in \mathbf{P}'$
- $s \in \iota_{init} \Rightarrow [s]_R \in \iota_{init}{}'$
- $\forall s \in S : L(s) \in L'([s]_R)$

3.3 Composition of Views

In essence views are simplification generated from an MDP. It seems rather obvious that the composition of views is a very practical feature. Therefore in this chapter we will introduce, formalize and discuss different notions of compositions. All variants will ensure that the effect caused by each partaking views also takes effect in the composed view. Moreover it is to be generated in a way that the effect of each individual view can be reverted from the composed one. There may be restrictions in the order of removal.

3.3.1 Parallel Composition

One of the most uncomplicated ideas is to group states that match in the function value of all given grouping function s. This idea is parallel in the sense that a set of grouping function is combined to a new grouping function in one single step.

Definition 3.4. Let $F_1, F_2, \ldots, F_n$ be grouping function s. A *parallel composed grouping function* is a grouping function $F_{\theta_{F_1||F_2||\ldots||F_n}} : S \to M, s \mapsto (F_1(s), F_2(s), \ldots, F_n(s))$.

A view of such a grouping function is as usual created in in accordance with Definition 3.3. Furthermore it is obvious that the effect of a view is considered in the new grouping function and hence the view. Since we define a new grouping function, the given ones and its views are retained.
The operator || used in the definition is derived from the operator used in electric circuits, when the respective elements are connected in parallel.
If we want to speak about this grouping function in a general way, where it is only of importance that we refer to this type of composition and the given grouping function s are of no importance, we will denote a *parallel composition grouping function* with $F_{||}$.

Other variants missing if existent Subsetapplication missing

4 View Examples

In this chapter we will introduce and discuss some view examples created by the author. Their purpose is to understand the idea and concept of a view and get to know some views that might be useful in real world applications.
When considering views we only want into account those that utilize properties of MDPs or that do computations that are also feasible on normal graphs but are of explicit relevance MDPs.

4.1 Views Utilizing MDP Components

In this subsection we will introduce some views the are purely based on the components of an MDP. Their will neither be computations on the graph-structure of an MDP nor computations using the result vector.

4.1.1 Atomic Propositions

One of the least involved approaches to create a view is to base it on the atomic propositions that can be assigned to each state by the labeling function. The notion is to group states that were assigned the same set of atomic propositions. Why useful?, currently no "has AP" -¿ maybe should be added

Definition 4.1. Let $\mathcal{M} = (S, Act, \mathbf{P}, \iota_{init}, AP, L)$ be an MDP. The view $\mathcal{M}_{AP}$ is defined by its grouping function F_{AP} grouping function with $F_{AP} : S \to M, s \mapsto L(s)$.

The grouping function is exactly the labeling function i.e. for all $s \in S$ it is $F_{AP}(s) = L(s)$. So it is $F_{AP}(s_1) = F_{AP}(s_2) \iff L(s_1) = L(s_2)$. According to Definition 3.2 for $\tilde{s} \in S$ it is $[\tilde{s}]_R = \{s \in S \mid L(s) = L(\tilde{s})\}$.

By this we obtain the view $\mathcal{M}_{AP}$ for a given an MDP $\mathcal{M}$ where: $S' = \bigcup_{s \in S}\{[s]_R\} = \bigcup_{a \in AP}\{\{s \in S \mid L(s) = a\}\}$. All other components are constructed as in Definition 3.3.

tikz example

4.1.2 Initial States

An a little more involved idea than directly using a given function is to utilize the set of initial states. We can group states that have a probability greater zero, that they are started from. In practice this might be useful to quickly find all initial states.

Definition 4.2. Let $\mathcal{M} = (S, Act, \mathbf{P}, \iota_{init}, AP, L)$ be an MDP and $I := \{s \in S \mid s \in \iota_{init}\}$. The view $\mathcal{M}_I$ is defined by its grouping function F_I grouping function with $F_I : S \to M$ with

$$s \mapsto \begin{cases} \emptyset, & \text{if } s \in I \\ s, & \text{otherwise} \end{cases}$$

and $M := \{\emptyset\} \cup S$.

For $s_1, s_2 \in S$ it is $F_I(s_1) = F_I(s_2)$ if and only if $s_1, s_2 \in I$ or $s_1 = s_2$. According to Definition 3.2 it is

$$\begin{aligned} [s]_R &= \{s \in S \mid F_I(s) = \emptyset\} && \text{for } s \in I \text{ and} \\ [s]_R &= \{s \in S \mid F_I(s) = \{s\}\} = \{s\} && \text{for } s \notin I. \end{aligned}$$

By this we obtain the view $\mathcal{M}_I$ for a given an MDP $\mathcal{M}$ where: $S' = \bigcup_{s \in S}\{[s]_R\} = \{s \in S \mid s \in I\} \cup \bigcup_{s \in S \setminus I}\{\{s\}\}$.

All other components are constructed as in Definition 3.3.

4.1.3 Outgoing Actions

define outgoing transition and outgoing action The *OutAction View* groups states that share some property regarding their actions of the outgoing transitions. Several variants are feasible. In the following we will use the expression "outgoing action α " equivalent with "transition with outgoing action α ".

The most obvious variant to group states is to group states that *have* a given outgoing action.

Definition 4.3. Let $\mathcal{M} = (S, Act, \mathbf{P}, \iota_{init}, AP, L)$ be an MDP and $\alpha \in Act$. The view $\mathcal{M}_{\exists \overrightarrow{\alpha}}$ is defined by its grouping function $F_{\exists \overrightarrow{\alpha}} : S \to M$ with

$$s \mapsto \begin{cases} \alpha, & \text{if } \exists s' \in S : (s, \alpha, s') \in \mathbf{P} \\ s, & \text{otherwise} \end{cases}$$

and $M := Act \cup S$.

For $s_1, s_2 \in S$ it is $F_{\exists \overrightarrow{\alpha}}(s_1) = F_{\exists \overrightarrow{\alpha}}(s_2)$ if and only if there exist $s_a, s_b \in S$ with $(s_1, \alpha, s_a), (s_2, \alpha, s_b) \in \mathbf{P}$ (i.e. they have the same outgoing action α) or $s_1 = s_2$. In accordance with Definition 3.2 it is

$$\begin{aligned} [s]_R &= \{s \in S \mid F_{\exists \overrightarrow{\alpha}}(s) = \alpha\} && \exists s' \in S : (s, \alpha, s') \in \mathbf{P} \\ [s]_R &= \{s \in S \mid F_{\exists \overrightarrow{\alpha}}(s) = s\} = \{s\} && \text{otherwise} \end{aligned}$$

Thereby we obtain the view $\mathcal{M}_{\exists \overrightarrow{\alpha}}$ for a given an MDP $\mathcal{M}$ where $S' = \bigcup_{s \in S}\{[s]_R\} =: S_1 \cup S_2$ where

$$\begin{aligned} S_1 &:= \{s \in S \mid \exists s' \in S : (s, \alpha, s') \in \mathbf{P}\} = \{s \in S \mid s \text{ has outgoing action } \alpha\} \\ S_2 &:= \bigcup_{s \in S \setminus S_1} \{\{s\}\}. \end{aligned}$$

Since actions are a very important part of MDPs as well as of its more powerful siblings MDPs and MCs it seems useful to further enhance this view and look at variants of it. Instead of only grouping states that only *have* outgoing actions we could also quantify the amount of times that action should be outgoing.

For example we could require that a given action has to be outgoing a minimum amount of times.

Definition 4.4. Let $\mathcal{M} = (S, Act, \mathbf{P}, \iota_{init}, AP, L)$ be an MDP and $\alpha \in Act$. The view $\mathcal{M}_{n\leq\overrightarrow{\alpha}}$ is defined by its grouping function $F_{n\leq\overrightarrow{\alpha}} : S \to M$ with

$$s \mapsto \begin{cases} \alpha, & \text{if } \exists s_1, \ldots, s_n \in S : Q_{n\leq\overrightarrow{\alpha}}(s, s_1, \ldots, s_n) \\ s, & \text{otherwise} \end{cases}$$

where $M := Act \cup S$, $n \in \mathbb{N}$ is the minimum amount of times a transition with action α has to be outgoing in order to be grouped with the other states and

$$Q_{n\leq\overrightarrow{\alpha}}(s, s_1, \ldots, s_n) := ((s, \alpha, s_1), \ldots, (s, \alpha, s_n) \in \mathbf{P}) \wedge |\{s_1, \ldots, s_n\}| = n$$

is a first order logic predicate.

The number and The predicate $Q_{n\leq\overrightarrow{\alpha}}(s, s_1, \ldots, s_n)$ requires that there are transitions with action α to n distinct states.

For $s_1, s_2 \in S$ it is $F_{n\leq\overrightarrow{\alpha}}(s_1) = F_{n\leq\overrightarrow{\alpha}}(s_2)$ if and only if there exist distinct $s_{a_1}, \ldots, s_{a_n} \in S$ and distinct $s_{b_1}, \ldots, s_{b_n} \in S$ so that $(s_1, \alpha, s_{a_1}), \ldots, (s_1, \alpha, s_{a_n}) \in \mathbf{P}$ and $(s_2, \alpha, s_{b_1}), \ldots, (s_2, \alpha, s_{b_n}) \in \mathbf{P}$ or $s_1 = s_2$. According to Definition 3.2 it is

$$\begin{aligned} [s]_R &= \{s \in S \mid F_{n\leq\overrightarrow{\alpha}}(s) = \alpha\} && \text{if } Q_{n\leq\overrightarrow{\alpha}}(s, s_1, \ldots, s_n) \\ [s]_R &= \{s \in S \mid F_{n\leq\overrightarrow{\alpha}}(s) = s\} = \{s\} && \text{otherwise} \end{aligned}$$

By this we obtain the view $\mathcal{M}_{n\leq\overrightarrow{\alpha}}$ for a given an MDP $\mathcal{M}$ where $S' = \bigcup_{s\in S}\{[s]_R\} =: S_1 \cup S_2$ where

$$\begin{aligned} S_1 &:= \{s \in S \mid \exists s_1, \ldots, s_n \in S : Q_{n\leq\overrightarrow{\alpha}}(s, s_1, \ldots, s_n)\} \\ &= \{s \in S \mid \text{the action } \alpha \text{ is outgoing at least } n \text{ times}\} \text{ and} \\ S_2 &:= \bigcup_{s\in S\setminus S_1} \{\{s\}\}. \end{aligned}$$

In a similar fashion we define view that groups states where at most a certain number of times a given action is outgoing.

Definition 4.5. Let $\mathcal{M} = (S, Act, \mathbf{P}, \iota_{init}, AP, L)$ be an MDP and $\alpha \in Act$. The view $\mathcal{M}_{\overrightarrow{\alpha}\leq n}$ is defined by its grouping function $F_{\overrightarrow{\alpha}\leq n} : S \to M$ with

$$s \mapsto \begin{cases} \alpha, & \text{if } \forall s_1, \ldots, s_{n+1} \in S : Q_{\overrightarrow{\alpha}\leq n}(s, s_1, \ldots, s_{n+1}) \\ s, & \text{otherwise} \end{cases}$$

where $M := Act \cup S$, $n \in \mathbb{N}$ is the maximal number of times a transition with action α may be outgoing and

$$Q_{\overrightarrow{\alpha}\leq n}(s, s_1, \ldots, s_{n+1}) := ((s, \alpha, s_1), \ldots, (s, \alpha, s_{n+1}) \in \mathbf{P}) \implies \bigvee_{\substack{i,j\in\{1,\ldots,n+1\} \\ i<j}} s_i = s_j$$

is a first order logic predicate.

It ensures that if there are one more than n outgoing transitions with an action α at least two of the states where the transitions end are in fact the same. Since this is required for all possible combinations of $n+1$ states by the grouping function, only states that have at most n outgoing actions will be assigned with α by the grouping function. The reasoning about the equality of the grouping function values, the obtained equivalence classes and the resulting set of states S' of the view is analogous to $\mathcal{M}_{n\leq\overrightarrow{\alpha}}$.

Since we already defined grouping functions and hence views for a required minimal and maximal amount of times an action has to be outgoing it is now easily possible to define a view that groups states where the amount of outgoing actions is at least n and at most m.

Definition 4.6. Let $\mathcal{M} = (S, Act, \mathbf{P}, \iota_{init}, AP, L)$ be an MDP and $\alpha \in Act$. The view $\mathcal{M}_{m\leq\overrightarrow{\alpha}\leq n}$ is defined by its grouping function $F_{m\leq\overrightarrow{\alpha}\leq n} : S \to M$ with

$$s \mapsto \begin{cases} \alpha, & \text{if } \exists s_1, \ldots, s_m \in S : Q_{m\leq\overrightarrow{\alpha}}(s, s_1, \ldots, s_m) \\ & \text{and } \forall s_1, \ldots, s_{n+1} \in S : Q_{\overrightarrow{\alpha}\leq n}(s, s_1, \ldots, s_{n+1}) \\ s, & \text{otherwise} \end{cases}$$

where $M := Act \cup S$ and $m, n \in \mathbb{N}$ are the minimal and maximal number of transitions with action α in order for state to be grouped. The predicates $Q_{n\leq\overrightarrow{\alpha}}(s, s_1, \ldots, s_n)$ and $Q_{\overrightarrow{\alpha}\leq n}(s, s_1, \ldots, s_{n+1})$ are the predicates from Definition 4.4 and Definition 4.5 respectively.

We already know that for a given $s \in S$ the expressions $\exists s_1, \ldots, s_n \in S : Q_{n\leq\overrightarrow{\alpha}}(s, s_1, \ldots, s_n)$ and $\forall s_1, \ldots, s_{m+1} \in S : Q_{\overrightarrow{\alpha}\leq m}(s, s_1, \ldots, s_{m+1})$ from Definition 4.4 and Definition 4.5 require that s has minimal and maximal amount of outgoing transitions with an action α respectively. Hence the conjunction will be true for states where the amount of outgoing transitions with action α is element of the set $\{m, n+1, \ldots, n-1, n\}$. We will write for this that the number of outgoing actions is *in the span*.

For a given state s and action α we set

$$C_{s,\overrightarrow{\alpha}} := \exists s_1, \ldots, s_m \in S : Q_{m\leq\overrightarrow{\alpha}}(s, s_1, \ldots, s_m) \wedge \forall s_1, \ldots, s_{n+1} \in S : Q_{\overrightarrow{\alpha}\leq n}(s, s_1, \ldots, s_{n+1})$$

for convenience. $C_{s,\overrightarrow{\alpha}}$ is true if and only if the number of outgoing actions is in the span. For $s_1, s_2 \in S$ it is $F_{m\leq\overrightarrow{\alpha}\leq n}(s_1) = F_{m\leq\overrightarrow{\alpha}\leq n}(s_2)$ if and only if $C_{s_1,\overrightarrow{\alpha}} \wedge C_{s_2,\overrightarrow{\alpha}}$ or $s_1 = s_2$. Then its equivalence classes are

$$\begin{aligned} [s]_R &= \{s \in S \mid F_{m\leq\overrightarrow{\alpha}\leq n}(s) = \alpha\} && C_{s,\overrightarrow{\alpha}} \text{ true} \\ [s]_R &= \{s \in S \mid F_{m\leq\overrightarrow{\alpha}\leq n}(s) = s\} = \{s\} && \text{otherwise} \end{aligned}$$

The new set of states S' of the view $\mathcal{M}_{m\leq\overrightarrow{\alpha}\leq n}$ is the union of the equivalence classes of equivalence relation R on the set of states S of the original an MDP. Hence it is $S' = \bigcup_{s\in S} [s]_R =: S_1 \cup S_2$ where

$$
\begin{aligned}
S_1 &:= \{s \in S \mid F_{m \leq \overrightarrow{\alpha} \leq n}(s) = \alpha\} \\
&= \{s \in S \mid C_{s,\overrightarrow{\alpha}} \text{ true}\} \\
&= \{s \in S \mid \text{the action } \alpha \text{ is outgoing } m \text{ to } n \text{ times}\} \text{ and} \\
S_2 &:= \bigcup_{s \in S \setminus S_1} \{\{s\}\}.
\end{aligned}
$$

The views above can be combined with parallel composition. The thereby obtained view requires that the respective conditions of all the combined views are met. In this sense it is a conjunctive combination.

Instead of making requirements about states and group them based on whether they meet these requirements it also possible to group states that are very similar or even identical in regard to their outaction. We consider this idea with the *OutActionsIdent View* in two variants: strong and weak (idententiy). Firstly we will consider the variant of strong identity.

Definition 4.7. Let $\mathcal{M} = (S, Act, \mathbf{P}, \iota_{init}, AP, L)$ be an MDP and $\alpha \in Act$. The view $\mathcal{M}_{\overrightarrow{Act}(s)_=}$ is defined by its grouping function $F_{\overrightarrow{Act}(s)_=} : S \to M$ with

$$s \mapsto \{(\alpha, n) \mid \alpha \in Act, n \text{ is the number of times that } \alpha \text{ is outgoing from } s\}$$

and $M := Act \times \mathbb{N}_0$.

The grouping function asserts to each state a set of pairs. Note that a pair is contained int the set for each action $\alpha \in Act$. In case there is no outgoing transition from state s with an action α it is $(\alpha, 0) \in F_{\overrightarrow{Act}(s)_=}$. For $s_1, s_2 \in S$ it is $F_{\overrightarrow{Act}(s)_=}(s_1) = F_{\overrightarrow{Act}(s)_=}(s_2)$ if and only if s_1 and s_2 are mapped to the same set of pairs. By Definition 3.2 the obtained equivalence classes of R are

$$[s]_R := \{s \in S \mid F_{\overrightarrow{Act}(s)_=}(s) = \{(\alpha_1, n_1), \ldots, (\alpha_l, n_l)\}, l = |Act|\}$$

According to Definition 3.3 the set $S' := \bigcup_{s \in S} [s]_R$ is the set of states of $\mathcal{M}_{\overrightarrow{Act}(s)_=}$. All other components of $\mathcal{M}_{\overrightarrow{Act}(s)_=}$ are as usual structured in accordance with the Definition 3.3. As mentioned earlier a weak variant of the OutActionsIdent view is also conceivable.

Definition 4.8. Let $\mathcal{M} = (S, Act, \mathbf{P}, \iota_{init}, AP, L)$ be an MDP and $\alpha \in Act$. The view $\mathcal{M}_{\overrightarrow{Act}(s)_\approx}$ is defined by its grouping function $F_{\overrightarrow{Act}(s)_\approx} : S \to M$ with

$$s \mapsto \{\alpha \in Act \mid \exists s' \in S : (s, \alpha, s') \in \mathbf{P}\}$$

and $M := Act$.

condition of image-set written in inconsistent style to strong identity. Swap strong and weak (order)?

The grouping function maps to the set of outgoing actions of a state and thereby discards information about the number of times an actions is outgoing. If an action is not outgoing from a state it is not contained in the set.

For $s_1, s_2 \in S$ it is $F_{\overrightarrow{Act(s)_\approx}}(s_1) = F_{\overrightarrow{Act(s)_\approx}}(s_2)$ if and only if they are mapped to the same set of actions. Hence the equivalence classes of R are

$$[\tilde{s}]_R = \{s \in S \mid F_{\overrightarrow{Act(s)_\approx}}(s) = F_{\overrightarrow{Act(s)_\approx}}(\tilde{s}) =: \{\alpha_1, \dots, \alpha_l\}, l \in \mathbb{N}\}.$$

According to Definition 3.3 the set $S' := \bigcup_{s \in S} [s]_R$ is the set of states of $\mathcal{M}_{\overrightarrow{Act(s)_\approx}}$.

4.1.4 Ingoing Actions

Analogously to Outgoing Actions views of utilizing ingoing actions are feasable. Since there is no difference apart from the definitions itself, we only provide the definitions.

Definition 4.9. Let $\mathcal{M} = (S, Act, \mathbf{P}, \iota_{init}, AP, L)$ be an MDP and $\alpha \in Act$. The view $\mathcal{M}_{\exists \overleftarrow{\alpha}}$ is defined by its grouping function $F_{\exists \overleftarrow{\alpha}} : S \to M$ with

$$s \mapsto \begin{cases} \alpha, & \text{if } \exists s' \in S : (s', \alpha, s) \in \mathbf{P} \\ s, & \text{otherwise} \end{cases}$$

and $M := Act \cup S$.

Definition 4.10. Let $\mathcal{M} = (S, Act, \mathbf{P}, \iota_{init}, AP, L)$ be an MDP and $\alpha \in Act$. The view $\mathcal{M}_{n \leq \overleftarrow{\alpha}}$ is defined by its grouping function $F_{n \leq \overleftarrow{\alpha}} : S \to M$ with

$$s \mapsto \begin{cases} \alpha, & \text{if } \exists s_1, \dots, s_n \in S : Q_{n \leq \overleftarrow{\alpha}}(s, s_1, \dots, s_n) \\ s, & \text{otherwise} \end{cases}$$

where $M := Act \cup S$, $n \in \mathbb{N}$ is the minimum amount of times a transition with action α has to be ingoing in order to be grouped with the other states and

$$Q_{n \leq \overleftarrow{\alpha}}(s, s_1, \dots, s_n) := ((s_1, \alpha, s), \dots, (s_n, \alpha, s) \in \mathbf{P}) \wedge |\{s_1, \dots, s_n\}| = n$$

is a first order logic predicate.

Definition 4.11. Let $\mathcal{M} = (S, Act, \mathbf{P}, \iota_{init}, AP, L)$ be an MDP and $\alpha \in Act$. The view $\mathcal{M}_{\overleftarrow{\alpha} \leq n}$ is defined by its grouping function $F_{\overleftarrow{\alpha} \leq n} : S \to M$ with

$$s \mapsto \begin{cases} \alpha, & \text{if } \forall s_1, \dots, s_{n+1} \in S : Q_{\overleftarrow{\alpha} \leq n}(s, s_1, \dots, s_{n+1}) \\ s, & \text{otherwise} \end{cases}$$

where $M := Act \cup S$, $n \in \mathbb{N}$ is the maximal number of times a transition with action α may be ingoing and

$$Q_{\overleftarrow{\alpha} \leq n}(s, s_1, \dots, s_{n+1}) := ((s_1, \alpha, s), \dots, (s_{n+1}, \alpha, s) \in \mathbf{P}) \implies \bigvee_{\substack{i,j \in \{1,\dots,n+1\} \\ i<j}} s_i = s_j$$

is a first order logic predicate.

Definition 4.12. Let $\mathcal{M} = (S, Act, \mathbf{P}, \iota_{init}, AP, L)$ be an MDP and $\alpha \in Act$. The view $\mathcal{M}_{m \leq \overleftarrow{\alpha} \leq n}$ is defined by its grouping function $F_{m \leq \overleftarrow{\alpha} \leq n} : S \to M$ with

$$s \mapsto \begin{cases} \alpha, & \text{if } \exists s_1, \ldots, s_m \in S : Q_{m \leq \overleftarrow{\alpha}}(s, s_1, \ldots, s_m) \\ & \text{and } \forall s_1, \ldots, s_{n+1} \in S : Q_{\overleftarrow{\alpha} \leq n}(s, s_1, \ldots, s_{n+1}) \\ s, & \text{otherwise} \end{cases}$$

where $M := Act \cup S$ and $m, n \in \mathbb{N}$ are the minimal and maximal number of transitions with action α in order for state to be grouped. The predicates $Q_{n \leq \overleftarrow{\alpha}}(s, s_1, \ldots, s_n)$ and $Q_{\overleftarrow{\alpha} \leq n}(s, s_1, \ldots, s_{n+1})$ are the predicates from Definition 4.9 and Definition 4.10 respectively.

Definition 4.13. Let $\mathcal{M} = (S, Act, \mathbf{P}, \iota_{init}, AP, L)$ be an MDP and $\alpha \in Act$. The view $\mathcal{M}_{\overleftarrow{Act(s)}_=}$ is defined by its grouping function $F_{\overleftarrow{Act(s)}_=} : S \to M$ with

$$s \mapsto \{(\alpha, n) \mid \alpha \in Act, n \text{ is the number of times that } \alpha \text{ is ingoing from } s\}$$

and $M := Act \times \mathbb{N}_0$.

Definition 4.14. Let $\mathcal{M} = (S, Act, \mathbf{P}, \iota_{init}, AP, L)$ be an MDP and $\alpha \in Act$. The view $\mathcal{M}_{\overleftarrow{Act(s)}_\approx}$ is defined by its grouping function $F_{\overleftarrow{Act(s)}_\approx} : S \to M$ with

$$s \mapsto \{\alpha \in Act \mid \exists s' \in S : (s', \alpha, s) \in \mathbf{P}\}$$

and $M := Act$.

4.1.5 Parameters

The concept of parameters is not part of the definitions of neither MDPs, MCs or MDPs. Even though, it is of great importance in practical applications. Parameters are used to represent states (in more detail).

For example an MDP could be used to model a computer program with human interaction. Every state of the MDP refers an overall state of the program during execution time. In this state of the program, its variables will have specific values. We may want to retain the information about the variable's values of the program instead of only assigning a state $s \in S$ that refers to the state of the program. Variables and its current values are not only relevant to computer programs but also other systems. Many of those other systems rely on some kind of global state during execution, which can be expressed with variables and values assigned to them. Since there exists no explicit component to retain this information the notion of parameters is used to store it.

OLD: Since transitions systems MCs and MDPs in practice are used to model, analyze and check real world systems it is very practical to not only name states but also describe the properties of the state in more detail. For a basic notion parameters are to be imagined as a set of variables that may have different values in different states thereby describing the characteristics of the state more thoroughly. ADDED TO OLD: In practice most often they arise naturally for example as variables of a computer program that is to be modeled with an MDP.

Because of the vast importance in practical applications we will consider some views that utilize them. To do so and being able to describe them formally we define and formalize the notion of parameters by considering them as a subset of the atomic propositions AP that is assigned a value by a function.

Definition 4.15. The set $Par \subseteq AP$ is called *parameters*.

Definition 4.16. Let M be an arbitrary set. The function $ParEval : S \times Par \to M$ is called parameter evaluation function.

Most of the time we will use $ParEval$ to refer to the parameter evaluation function. When we speak about the value of a parameter in a state we refer to the image of $ParEval$ for that state and parameter. The set M is arbitrary so that arbitrary values can be assigned to a parameter. Speaking in terms of computer science and programming this loosens as an example the restriction of only being able to assign numbers and no booleans.

The most apparent idea for a view utilizing parameters is to group states that meet some requirement regarding the values of the parameters.

state has parameter not here uptil now because probably not used

Definition 4.17. Let

- $\mathcal{M} = (S, Act, \mathbf{P}, \iota_{init}, AP, L)$ be an MDP,
- $x \in Par \subseteq AP$ and
- $ParEval(s, x) = a$ where $s \in S$.

The view $\mathcal{M}_{x=a}$ is defined by its grouping function $F_{x=a} : S \to M$ with

$$s \mapsto \begin{cases} a, & \text{if } ParEval(s, x) = a \\ s, & \text{otherwise} \end{cases}$$

where $M := S \cup \{a\}$.

The view $\mathcal{M}_{x=a}$ groups states that share the same value for a given parameter. For s_1, s_2 it is $F_{x=a}(s_1) = F_{x=a}(s_2)$ if and only if $ParEval(s_1, x) = ParEval(s_2, x)$ or $s_1 = s_2$. The obtained equivalence classes are

$$[s]_R = \{s \in S \mid ParEval(s, x) = a\}$$
$$[s]_R = \{s \in S \mid F_{x=a}(s) = s\} = \{s\}$$

The set of states S' of $\mathcal{M}_{x=a}$ is the union of the equivalence classes of R. It is $S' = \bigcup_{s \in S} [s]_R =: S_1 \cup S_2$ where

$$\begin{aligned} S_1 &:= \{s \in S \mid F_{x=a}(s) = a\} \\ &= \{s \in S \mid ParEval(s, x) = a\} \text{ and} \\ S_2 &:= \bigcup_{s \in S \setminus S_1} \{\{s\}\}. \end{aligned}$$

Analogously a view that requires inequality instead of equality is feasible.

Definition 4.18. Let

- $\mathcal{M} = (S, Act, \mathbf{P}, \iota_{init}, AP, L)$ be an MDP,
- $x \in Par \subseteq AP$ and
- $ParEval(s, x) \neq a$ where $s \in S$.

The view $\mathcal{M}_{x \neq a}$ is defined by its grouping function $F_{x \neq a} : S \to M$ with

$$s \mapsto \begin{cases} a, & \text{if } ParEval(s, x) \neq a \\ s, & \text{otherwise} \end{cases}$$

where $M := S \cup \{a\}$.

If states are to be grouped with the requirement of several parameters equaling or not equaling specified values this can be achieved by using parallel composition.

To allow even more flexibility a view can be used that also allows a combination of requirements on parameters in a disjunctive manner. To extend this idea to its full potential we will define a view that allows requirements using a disjunctive normal formal (DNF). To formalize this view more efficiently we will write $x_{s,kl} = a$ short for $ParEval(s, x_{kl}) = a$ where $k, l \in \mathbb{N}$ and $x_{kl} \in Par$. In the same sense we write $x_{s,kl} \neq a$. It may be that $x_{ab} = x_{mn}$ for $a, b, m, n \in \mathbb{N}$ NOT HAPPY WITH THIS. We define the symbol $\doteq$ to be an element of the set $\{=, \neq\}$. That is to say, whenever it is used each time written it is a representative for either the symbol $=$ or $\neq$. It allows to write one symbol whenever $=$ and $\neq$ could or should be possible. Moreover for this context we consider $(x_{s,kl} = a)$ as a literal and $(x_{s,kl} \neq a)$ as its negation. We write $(x_{s,kl} \doteq a)$ for a literal that could be negated or not negated.

Definition 4.19. Let $\mathcal{M} = (S, Act, \mathbf{P}, \iota_{init}, AP, L)$ be an MDP and

$$d(s) = ((x_{s,11} \doteq a_{11}) \wedge \cdots \wedge (x_{s,1l_1} \doteq a_{1l_1})) \vee \cdots \vee ((x_{s,k1} \doteq a_{k1}) \wedge \cdots \wedge (x_{s,kl_k} \doteq a_{kl_k}))$$

proposition logical formulae in disjunctive normal form where

- $\{x_{kl_i} \mid k \in \mathbb{N}, l_i \in \{l_1, \ldots l_k\} \subseteq \mathbb{N}\} \subseteq Par$ and $x_{s,kl_i} = ParEval(s, x_{kl_i})$
- $\{a_{kl_i} \mid k \in \mathbb{N}, l_i \in \{l_1, \ldots l_k\} \subseteq \mathbb{N}\} \subseteq ParEval(S, Par)$

The view $\mathcal{M}_{ParDNF}$ is defined by its grouping function $F_{ParDNF} : S \to M$ where

$$s \mapsto \begin{cases} true, & \text{if } d(s) \text{ is true} \\ s, & \text{otherwise} \end{cases}$$

where $M := \{true, false\}$.

The DNF allows us to specify a requirement in disjunctive normal form about parameters. States are mapped to the same value depending on whether or not they meet this requirement.

DISCUSSION OF EQUALITY, EQ CLASSES AND RESULTING STATES MISSING

Analogously a view based on a conjunctive normal formal can be defined that may be more convenient, depending on the query.

Definition 4.20. Let $\mathcal{M} = (S, Act, \mathbf{P}, \iota_{init}, AP, L)$ be an MDP and

$$c(s) = ((x_{s,11} \doteqdot a_{11}) \vee \cdots \vee (x_{s,1l_1} \doteqdot a_{1l_1})) \wedge \cdots \wedge ((x_{s,k1} \doteqdot a_{k1}) \vee \cdots \vee (x_{s,kl_k} \doteqdot a_{kl_k}))$$

proposition logical formulae in disjunctive normal form where

- $\{x_{kl_i} \mid k \in \mathbb{N}, l_i \in \{l_1, \dots l_k\} \subseteq \mathbb{N}\} \subseteq Par$ and $x_{s,kl_i} = ParEval(s, x_{kl_i})$
- $\{a_{kl_i} \mid k \in \mathbb{N}, l_i \in \{l_1, \dots l_k\} \subseteq \mathbb{N}\} \subseteq ParEval(S, Par)$

The view $\mathcal{M}_{ParCNF}$ is defined by its grouping function $F_{ParCNF} : S \to M$ where

$$s \mapsto \begin{cases} true, & \text{if } c(s) \text{ is true} \\ s, & \text{otherwise} \end{cases}$$

where $M := \{true, false\}$.

The only difference from the view $\mathcal{M}_{ParCNF}$ to the view $\mathcal{M}_{ParDNF}$ is whether the respective formulae is in conjunctive or disjunctive normal form.CITATION Since each formulae in conjunctive normal form can tan be transformed to a formulae in disjunctive normal form and vice verse neither on of the views can perform an action the other can not. Hence there is no difference in expressivity, but there may be one in size. Therefore both views have been implemented and formalized.

The views discussed before reduce the an MDP in a very precise but also manual manner, because it not only dictates the parameter but also its value. A more general approach is to stipulate only the parameter but not its value. This way states will be grouped that have the same value for that parameter with no regard to the actual value of that parameter. This idea could be achieved with a view based on the grouping function $F_{\theta F_{x=a_1} || \dots || F_{x=a_n}}$ with $ParEval(S, x) = \{a_1, \dots, a_n\}$ and $|ParEval(S, x)| = n$. This grouping function just groups on every possible value for the parameter x. Since this is not very practical we define a view that achieves this result in a more direct and more efficient way.

Definition 4.21. Let $\mathcal{M} = (S, Act, \mathbf{P}, \iota_{init}, AP, L)$ be an MDP. The view $\mathcal{M}_{ParEval(s,x)}$ is defined by its grouping function $F_{ParEval(s,x)} : S \to M$ with

$$s \mapsto ParEval(s, x)$$

and $M := ParEval(S, x)$.

With this grouping function we directly map to the value of the parameter. Hence for $s_1, s_2 \in S$ it is $F_{ParEval(s,x)}(s_1) = F_{ParEval(s,x)}(s_2)$ if and only if they are mapped to the value $a \in M$. Hence the equivalence classes of R are

$$[\tilde{s}]_R = \{s \in S \mid ParEval(s) = ParEval(\tilde{s})\}.$$

According to Definition 3.3 the set $S' := \bigcup_{s \in S} [s]_R$ is the set of states of $\mathcal{M}_{ParEval(s,x)}$.

4.2 Utilizing the MDP Graphstructure

4.2.1 Distance

TODO

Definition 4.22. Let $\mathcal{M} = (S, Act, \mathbf{P}, \iota_{init}, AP, L)$ be an MDP and $\tilde{S} \subseteq S$ arbitrary. *distance*$(\mathcal{M}, \tilde{S}, n)$ IMPLEMENTATION IN PSEUDOCODE? DEFINITION BEFORE? The view $\mathcal{M}_{\longleftrightarrow}$ is defined by its grouping function $F_{\longleftrightarrow} : S \to M$ with

$$s \mapsto dist \qquad \text{where } (s, dist) \in distance(\mathcal{M}, \tilde{S}, n)$$

and $M = S \times (\mathbb{N} \cup \{\inf\})$.

Due to the implementation of $distance(\mathcal{M}, \tilde{S}, n)$ for every state $s \in S$ exists a pair $(s, dist)$ in the returned set of $distance(\mathcal{M}, \tilde{S}, n)$. That is it is there for every state $s \in S$ there exists $(s, dist)$ with $(s, dist) \in distance(\mathcal{M}, \tilde{S}, n)$. The view groups states that have the same distance to the set measured with the amounts of transitions necessary reach the the closest $\tilde{S}$ considering the granularity n.

4.2.2 Double Directed Edges with same Action

NOT SURE IF SHOULD BE INCLUDED

Definition 4.23. Let $\mathcal{M} = (S, Act, \mathbf{P}, \iota_{init}, AP, L)$ be an MDP.

4.2.3 Cycles

A cycle is a structure that can exist in every graph. The concept of cycles is not specific to MDPs or any of its more specialized variants. The purpose of this thesis is to discuss views that utilize domain specific knowledge or if a general concept is of special relevance when exploring an MDP. The former and the latter apply on cycles.

Formalizing views based on cycles requires some formalization of the concept cycle. We will use a domain specific definition that will serve us the most.

Definition 4.24. Let $\mathcal{M} = (S, Act, \mathbf{P}, \iota_{init}, AP, L)$ be an MDP. A (simple) *cycle* in $\mathcal{M}$ is any sequence $(s_0, \alpha_0, s_1, \alpha_1, \ldots, \alpha_{n-1}, s_0)$ alternating between states and actions where $n \in \mathbb{N}, \{s_0, \ldots, s_{n-1}\} \subseteq S$ is a set of distinct states, $\{\alpha_0, \ldots, \alpha_{n-1}\} \subseteq Act$ and for all $i \in \{0, \ldots, n-1\}$ it is $(s_i, \alpha_i, s_{i+1 \bmod n}) \in \mathbf{P}$.

When the actions in the cycle are of no further importance, we will omit them only writing a sequence of states. In the following let $C = (s_0, \alpha_0, s_1, \alpha_1, \ldots, \alpha_{n-1}, s_0)$ be a cycle. For conveniences we will write $s \in C$ if the state is contained in the cycle C and $\alpha \in C$ if the action is contained in in the cycle C. only if?? In words we will write a state or action is *on* or *in* the cycle.Let C_1 and C_2 be cycles. $C_1 \cup C_2 := \{s \in S \mid s \in C_1 \text{ or } s \in C_2\}$.

Definition 4.25. Let C be an cycle in $\mathcal{M}$ and S be the states set of $\mathcal{M}$. The number $n = |\{s \in S \mid s \in C\}|$ is called the *length* of a cycle.

Apart from the formalization of the term cycle and the length of a cycle we will also have to formalize a way to find cycles.

Definition 4.26. Let $\mathcal{M} = (S, Act, \mathbf{P}, \iota_{init}, AP, L)$ be an MDP and $n \in \mathbb{N}$. The function $findCycles(\mathcal{M}, n)$ is called *cycle finder function* (CFF). It returns a set $C_{\{\}}$ of all cycles in $\mathcal{M}$ where for all $C \in C_{\{\}}$ it is $|\{s \in S \mid s \in C\}| \geq n$. That is there are at least n states contained in each cycle.

In a programming like manner we will write $findCycles(\mathcal{M}, n)$ and thereby referring to its returned set. This will be clear from the context.

In practice there exist several cycle finding algorithms. The function $findCycles(\mathcal{M}, n)$ is an abstraction for one of these algorithms being used. The actual implementation relys on algorithms of the jave library jgrapht namely the Algorithm Szwarcfiter and Lauer - $O(V + EC)$ and Tiernan - $O(V.constV)$ CITATION!!

With the formalization done we will introduce some views utilizing the concept cycles. For one we will combine the notion cycle with domain specific knowledge for the other we will consider how and why cycles in MDPs are in general of relevance. We will begin with the latter. Cycles in general are of interest because they pose the risk of getting stuck in endless loops, when performing actions on the MDP. Model checking is one of the most relevant actions on MDPs that are vulnerable to loops. They are performed commonly on them. Therefore a view that simply finds existing cycles is feasible.

Definition 4.27. Let $\mathcal{M} = (S, Act, \mathbf{P}, \iota_{init}, AP, L)$ be an MDP and $n \in \mathbb{N}$. The view $\mathcal{M}_{\exists C}$ is defined by its grouping function $F_{\exists C} : S \to M$ with

$$s \mapsto \begin{cases} inCycle, & \text{if } findCycles(\mathcal{M}, n) \neq \emptyset \\ s, & \text{otherwise} \end{cases}$$

and $M = S \cup \{inCycle\}$.

This view groups all states that are contained in cycle with a length of at least n. It is to note that the affinity of a state to one or several respective cycles is lost.

DISCUSSION OF EQUALITY, EQ CLASSES AND RESULTING STATES MISSING

Definition 4.28. Let $\mathcal{M} = (S, Act, \mathbf{P}, \iota_{init}, AP, L)$ be an MDP and $n \in \mathbb{N}$. The view $\mathcal{M}_{\{C\}}$ is defined by its grouping function $F_{\{C\}} : S \to M$ with

$$s \mapsto \{C \in findCycles(\mathcal{M}, n) \mid s \in C\}$$

and $M = \mathcal{P}(findCycles(\mathcal{M}, n))$

This view groups states that have the same set of cycles they are contained in. Thus if C_1 and C_2 are distinct cycles and $s_1, s_2 \in C_1$ and $s_1 \in C_2$ but $s_2 \notin C_2$ they are not grouped. It suffices that a state is on one cycle that the other one is not, in order for the states not being grouped. In graphs with many cycles this can lead to little grouping.

DISCUSSION OF EQUALITY, EQ CLASSES AND RESULTING STATES MISSING

GREEDY APPROACHES (STATE ON SINGLE CYCLE), AND SET APPROACH s -¿ $C_1 \cup C_2$ omitted because not sure if needed

The view above might be useful when having found cycles to see what cycles specifically exist. Often it is interesting to find cycles that consist only of transitions of the same action. The following view accomplishes that.

Definition 4.29. Let $\mathcal{M} = (S, Act, \mathbf{P}, \iota_{init}, AP, L)$ be an MDP and $n \in \mathbb{N}$. The view $\mathcal{M}_{\{C\}}$ is defined by its grouping function $F_{\{C\}} : S \to M$ with

$$s \mapsto \{C \in findCycles(\mathcal{M}, n) \mid s \in C, \tilde{\alpha} \in C, \forall \alpha \in C : \alpha = \tilde{\alpha}\}$$

and $M = \mathcal{P}(findCycles(\mathcal{M}, n))$

The view specializes the view from Definition 4.28 in the sense that it additionally requires for all $C \in F_{\{C\}}$ that all actions occurring in the cycle are the same. doubling to above definition?

NO DISCUSSION IF PREVIOUSLY

4.2.4 Strongly Connected Components

Strongly connect components (SCC) are of major importance in model checking I think so because I found the terms many times in the book. Remaining question: really? why? Therefore it is feasible to consider a view utilizing strongly connected components. Deviating from most definitions we will define SCC not as a subgraph but only as the set of its nodes. Moreover the definition written in the terms of an MDP.

Definition 4.30. Let $\mathcal{M} = (S, Act, \mathbf{P}, \iota_{init}, AP, L)$ be an MDP. A set of the from

$$T = \{s_1 \in S \mid \forall s' \in T : \exists n \in \mathbb{N}, \exists (s_1, \alpha_1, s_2, \alpha_2 \ldots, s_n, \alpha_n, s') \\ \text{where } (s_n, \alpha_n, s') \in \mathbf{P} \wedge \forall i \in \{1, \ldots, n-1\} : (s_i, \alpha_i, s_{i+1}) \in \mathbf{P}\}$$

is called *strongly connected component* of $\mathcal{M}$. That is for every state in the set there has to exist a sequence of transitions to every other state of the set. The set of strongly connected components of $\mathcal{M}$ is denoted with $SCC(\mathcal{M})$.

Since the strong connection is an equivalence relation the SCCs are equivalence classes and hence disjoint. To find SCCs Tarjans algorithm is the classic. In the implementation an improved variant from Gabow is used supplied by the jGrapht-Tlibrary. CITATION

Definition 4.31. Let $\mathcal{M} = (S, Act, \mathbf{P}, \iota_{init}, AP, L)$ be an MDP. The view $\mathcal{M}_{scc}$ is defined by its grouping function $F_{scc} : S \to M$ with

$$s \mapsto T \qquad \text{where } s \in T \in SCC(\mathcal{M})$$

and $M = SCC(\mathcal{M})$.

This view groups all states together that are in the same SCC. Because SCCs are disjoint each s will be mapped to its one and only SCC. This is because... DISCUSSION OF EQUALITY, EQ CLASSES AND RESULTING STATES MISSING

A special kind of strongly connected components is the the bottom strongly connected component.

Definition 4.32. Let T be a SCC. A *bottom strongly connected component* (BSCC) is a SCC where it holds that: concurrent versions

$$\forall s \in T : \forall (s, \alpha, s') \in \mathbf{P} : s' \in T$$

$$\forall s \in T : \sum_{t \in T} \mathbf{P}(s, t) = 1$$

That is from T there is no state reachable outside of T. The set of bottom strongly connected components of $\mathcal{M}$ is denoted with $BSCC(\mathcal{M})$.

These are of special relevance because..

Definition 4.33. Let $\mathcal{M} = (S, Act, \mathbf{P}, \iota_{init}, AP, L)$ be an MDP. The view $\mathcal{M}_{bscc}$ is defined by its grouping function $F_{bscc} : S \to M$ with

$$s \mapsto T \qquad \text{where } s \in T \in BSCC(\mathcal{M})$$

and $M = BSCC(\mathcal{M})$.

In the implementation strongly connected components are determined using the algorithm of Gabow, afterwards filtering those SCCs that have only transitions to states within the SCC. Equality, equivalence classes and the new state set are constructed analogously to the view $\mathcal{M}_{scc}$.

4.3 Utilizing the MDP Result Table

In this section we will discuss views utilizing the result table. The actual implementation is relies on one powerful view that can set to perform arbitrary actions using model checking results.

5 View Implementation

supposed to give a short overview how view are implemented from a conceptional perspective. No more than 1 to 2 pages.

6 Comparison and Evaluation

Performance Cycles Clustering exact Cycles when clustering exact cycles

7 Outlook

Many other views accomplishable with the properties view. AP Cluster has Cycle All implemented Clusters in Appendix (only definition) $\mathbb{Z}_n$ $[s]_R$ $\mathbb{N}$ in symbol list Implementation of own algorithms Distance Cluster Forward backward both $\circlearrowleft$

References