
Counterexamples for Expected Rewards on Discrete-time Markov Chains

by
Tim Quatmann

Bachelor Thesis at RWTH Aachen University,
Lehrstuhl für Informatik 2

Submitted to: Fakultät für Mathematik, Informatik und
Naturwissenschaften der RWTH Aachen

Registration Date: July 11, 2014
Submission Date: September 29, 2014

First examiner: Prof. Dr. Ir. Joost-Pieter Katoen
Second examiner: Prof. Dr. Erika Ábrahám
Thesis adviser: Nils Jansen, Christian Dehnert

I hereby declare that I have created this work completely on my own and used no other sources or tools than the ones listed, and that I have marked any citations accordingly.

Hiermit versichere ich, dass ich die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt, sowie Zitate kenntlich gemacht habe.

Aachen, September 29, 2014

Tim Quatmann

Abstract

Whenever a system does not meet a given requirement, it is useful to indicate undesired behavior in terms of a counterexample. For a probabilistic system which is modeled as a discrete-time Markov chain, such a counterexample usually has to evidence enough probability. In recent work, so-called critical subsystems have been proposed to serve as counterexamples for reachability properties. In this context, a critical subsystem is a restriction of the original system such that a given set of states will already be reached with a certain probability.

The expressiveness of a discrete-time Markov chain can strongly be extended by assigning rewards which can be interpreted as time, attempts, steps, and the like. We can check properties regarding the expected rewards to ensure, for instance, that the expected number of required attempts is within a certain bound. A counterexample for this class of properties should indicate that at least a certain reward can be expected. Up to our knowledge, there is no work concerning counterexamples for expected rewards.

In this thesis, the idea of critical subsystems is applied to form counterexamples for expected rewards on discrete-time Markov chains. Besides a formal definition, the focus lies on the efficient generation of preferably small critical subsystems. For this purpose, two new heuristic approaches based on path search and best first search algorithms are introduced.

Überblick

Wenn ein System eine gegebene Bedingung nicht erfüllt, so ist es hilfreich unerwünschtes Verhalten in Form eines Gegenbeispiels kenntlich zu machen. Für ein probabilistisches System, welches durch eine Markow-Kette mit diskreter Zeit modelliert ist, muss solch ein Gegenbeispiel für gewöhnlich ausreichend Wahrscheinlichkeit bezeugen. In aktuellen Arbeiten wurden sogenannte kritische Subsysteme vorgeschlagen, um als Gegenbeispiele für Erreichbarkeitseigenschaften zu dienen. In diesem Kontext ist ein kritisches Subsystem eine Einschränkung des Originalsystems, so dass eine gegebene Zustandsmenge bereits mit einer gewissen Wahrscheinlichkeit erreicht werden kann.

Die Ausdrucksstärke von Markow-Ketten mit diskreter Zeit kann erheblich gesteigert werden, indem Kosten zugewiesen werden. Diese Kosten können als Zeit, Versuche, Schritte und Ähnliches interpretiert werden. Wir können Eigenschaften bezüglich der erwarteten Kosten überprüfen um beispielsweise sicherzustellen, dass die erwartete Anzahl an Versuchen innerhalb einer gewissen Schranke liegt. Ein Gegenbeispiel für diese Art von Eigenschaften sollte bezeugen, dass mindestens eine gewisse Menge von Kosten erwartet werden können. Nach unserem Kenntnisstand gibt es keine Arbeit welche sich mit Gegenbeispielen für die erwarteten Kosten befasst.

In dieser Arbeit wird die Idee von kritischen Subsystemen angewendet, um Gegenbeispiele für die erwarteten Kosten auf Markow-Ketten mit diskreter Zeit zu bilden. Neben einer formalen Definition liegt der Fokus auf der effizienten Generierung von vorzugsweise kleinen kritischen Subsystemen mithilfe heuristischer Verfahren. Zu diesem Zweck werden zwei neue heuristische Verfahren basierend auf Pfadsuche und Bestensuche vorgestellt.

Contents

1	Introduction	1
2	Preliminaries	5
2.1	Discrete-time Markov Chains with Rewards	5
2.2	Reachability Probabilities and Expected Rewards	8
2.3	Restrictions	12
3	Related Work	17
3.1	Path-based Counterexamples for Reachability Properties	17
3.2	Critical Subsystems for Reachability Properties	18
3.3	Tools and Implementations	24
4	Critical Subsystems Applied to Expected Reward Properties	25
4.1	Reachability of Target States	25
4.2	Critical Subsystems for Expected Reward Properties	26
4.3	Alternative Definition of Critical Subsystems	27
4.4	Reduction from Reachability Properties	28
5	Generating Critical Subsystems	33
5.1	Path Search Approach	33
5.1.1	The Basic Algorithm	33
5.1.2	Modifications of the Algorithm	39
5.1.3	Performance Observations and Improvements	41
5.2	Best-first Search Approach	41
5.2.1	The Algorithm	42
5.2.2	Different Value Functions	43
6	Experimental Results	47
6.1	Models and Properties for the Experiments	47
6.2	Results for Expected Reward Properties	48
6.3	Results for Reachability Properties	54
7	Conclusion and Future Work	57

Chapter 1

Introduction

Verifying the correctness and eliminating undesired behavior of a system is a very extensive task, especially if the system is complex and safety relevant. *Model checking* [5] is a well-known technique to facilitate this task. The idea is that the system is represented as a model for which a model checker can check whether a given property holds on it. If the property is refuted, a counterexample can often be provided to indicate the erroneous parts of the system. However, the exact behavior of a system is not always predictable as it may depend on unreliable channels, user inputs, random choices and the like. When it comes to modeling such systems, the classic approach is to use nondeterminism to formalize all possible cases.

For instance, let us consider a simple communication protocol where a station attempts to send a message via an unreliable channel. If the delivery of the message is acknowledged, the sending process ends successful. Otherwise, the message will be send again. After three unsuccessful attempts, the sending process ends erroneous. If we model the unreliability of the channel in terms of nondeterminism, a model checker can answer questions like “Is it possible that the message will never be delivered?”. If the answer to this question is “yes”, the likelihood of a delivery failure is still unknown. The more interesting questions might be “Is the probability that a message will never be delivered within a certain bound?”. Another question of interest might be “Is the expected number of required attempts until delivery within a certain bound?”. In order to answer this questions, we also need to model the probability of losing a single message as well as the number of required attempts.

Discrete-time Markov chains with rewards. *Discrete-time Markov chains* are a well-known formalism that can be used to model *probabilistic systems*. In this work, we will adhere to the transition system perspective like it is presented in [4]. Hence, a discrete-time Markov chain (DTMC) can be seen as a transition system where the transitions are labeled with probabilities. The computation starts in one of the initial

states, according to a given initial distribution. If there is a transition labeled with probability p from the current state to a state s , then, with probability p , the next state will be s . Thus, the successive state is always given by a probabilistic choice between the outgoing transitions of the current state.

The states and transitions of a DTMC can also be equipped with rewards. Such an extended DTMC is called *Markov reward model* (MRM). The assigned reward is earned on leaving the corresponding state and on taking the corresponding transition, respectively. With an MRM, it is possible to model, for instance, power consumption, time consumption, manufactured items, or required attempts. A model checker can check properties concerning the probabilities and the rewards of a given MRM. An important example is the expected reward property. Roughly, the expected reward is the expected value of the cumulative reward on a path from the current state to one of the given target states. An expected reward property holds whenever the expected reward is within a certain bound. Moreover, we want to consider reachability properties which state whether the probability to reach a given set of states is within a certain bound. There are step-bounded and reward-bounded variants of reachability properties which refer to the probability to reach a set of states within a certain number of steps or without exceeding a certain amount of cumulated reward.

Figure 1.1 depicts an MRM that models the sender in the simple communication protocol described above. Initial states are marked with an arrow. As there is only one initial state (a_1), the initial distribution is clear and the computation will start in this state with probability 1. The states a_i for $i \in \{1, 2, 3\}$ represent the three attempts to send the message. The possible message loss is modeled in terms of a probabilistic choice. We assume that the channel works properly with probability 0.8. In this case, the system moves to the state *del* to indicate that the message has successfully been delivered. Otherwise, the message got lost and the sender performs the next attempt. After the third unsuccessful attempt, the sending process ends erroneous and the system moves to the state *err*. The states of an MRM (or a DTMC) can be labeled with sets of atomic propositions depicted by a set next to the state. These propositions are helpful to formally define properties for the model. In this example, the state(s) where the message has successfully been delivered are labeled with **success**, the state(s) where the protocol run ends unsuccessful are labeled with **failure**, and the state(s) where the protocol run ends are labeled with **end**. We want to consider the expected number of attempts for a protocol run. For this purpose, rewards are added to the model and interpreted as sending attempts. More precisely, all states where a message delivery is attempted (namely the states a_1 , a_2 , and a_3) are equipped with reward 1, depicted by a number next to these states. The rewards assigned to the remaining states and the transitions equate to 0 and are therefore not depicted. With this atomic propositions and reward assignments, the probability to reach a state labeled with **failure** corresponds to the probability that the message will never be delivered. Moreover, the expected reward until reaching a state labeled with **end** equals the expected number of attempts for a protocol run.

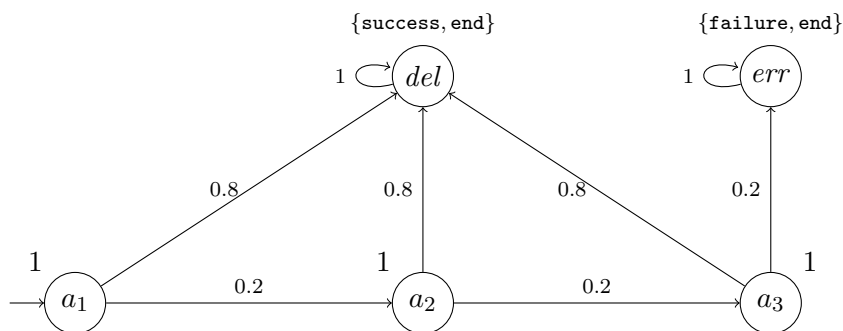


Figure 1.1: Markov reward model of a simple communication protocol.

Counterexamples. Whenever a property is violated, knowing which part of the model leads to the violation is crucial for fixing the problem. Therefore, a *counterexample* that indicates undesired behavior is very helpful. Generating counterexamples for reachability properties on discrete-time Markov Chains is a very active research field. An overview of recent results is given in [1].

We typically consider reachability properties with an upper bound on the probability to reach one of the target states. A counterexample for this kind of property has to indicate that the probability exceeds the given bound. It is helpful to determine a path that shows how a target state can be reached with high probability. In general, such a single path might not induce enough probability and is not sufficient to form a counterexample. In [8], a method to generate counterexamples as a set of paths leading to target states is presented. This method is applicable for reachability properties as well as step-bounded or reward-bounded reachability properties. However, the number of required path can be very high or even infinite. An alternative approach is the generation of a critical subsystem. The idea is to restrict the original system to a preferably small subsystem that itself induces enough probability to violate the property. There are several approaches to generate critical subsystems for reachability properties (e.g. [2, 11, 19]), resulting in subsystems where the considered states form only a small subset of the original state space.

Counterexamples for expected reward properties with an upper bound have to indicate that the expected reward until reaching one of the target states exceeds a certain value. Similar to counterexamples for reachability properties, a single path might not suffice for this purpose. To our knowledge, there is no work concerning counterexamples for expected reward properties.

In this thesis we will discuss how the notion of critical subsystems can be applied to form counterexamples for expected reward properties. Beside the theoretical foundations, we will examine two approaches that can be used to generate critical subsystems for this class of properties. These approaches efficiently find and select states that are promising to be part of a small critical subsystem. The generation stops as soon as the selected

states induce a critical subsystem. The path search approach uses a modification of a shortest path algorithm to search for paths with a good evaluation. The best-first search approach considers the states adjacent to already selected states and repeatedly selects the most promising of these states.

Structure of this thesis. In Chapter 2, we formally define MRMs and the properties we are going to consider. After that, we have a look on related work concerning counterexample generation for reachability probabilities in Chapter 3. In Chapter 4, we apply the notion of critical subsystems for expected rewards. Chapter 5 is dedicated to the generation of such critical subsystems introducing two heuristic approaches based on path search and best-first search algorithms. The experimental results will be discussed in Chapter 6. Finally, Chapter 7 concludes the presented topics and gives an outlook on future work.

Chapter 2

Preliminaries

In this chapter, we formally define discrete-time Markov chains equipped with rewards, which are called Markov reward models. After that, we specify reachability probabilities and expected rewards as well as the associated state-properties. Finally, we discuss reasonable restrictions to the general definitions. More information about the given definitions can be found in [4].

2.1 Discrete-time Markov Chains with Rewards

If we want to model a probabilistic system, we need a modeling formalism that takes the likelihood of the different behaviors of the system into account. For this purpose we make use of discrete-time Markov chains.

Definition 2.1 (Discrete-time Markov Chain)

A discrete-time Markov chain (DTMC) is a tuple $\mathcal{D} = (S, \mathbf{P}, \nu_{init}, AP, L)$ where

- S is a countable set of *states* with $S \neq \emptyset$,
- $\mathbf{P}: S \times S \rightarrow [0, 1]$ is a *transition probability function* where for all states $s \in S$ it holds that

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

- $\nu_{init}: S \rightarrow [0, 1]$ is an *initial distribution* such that

$$\sum_{s \in S} \nu_{init}(s) = 1,$$

- AP is a set of *atomic propositions*, and
- $L: S \rightarrow 2^{AP}$ is a *labeling function*. ■

In the following, consider a DTMC $\mathcal{D} = (S, \mathbf{P}, \iota_{init}, AP, L)$.

\mathcal{D} is called *finite*, if the sets S and AP are finite. Intuitively, \mathcal{D} starts in a state $s \in S$ with probability $\iota_{init}(s)$. The subsequent states are given by \mathbf{P} since the value $\mathbf{P}(s, s')$ coincides with the probability to move from state $s \in S$ to state $s' \in S$. We say that there is a *transition* from s to s' if and only if $\mathbf{P}(s, s') > 0$. We call s an *absorbing state* if $\mathbf{P}(s, s) = 1$ and $\mathbf{P}(s, s') = 0$ for all $s' \neq s$. Moreover, we use graph-like notions like successor or reachability, which refer to the graph $\mathcal{G}_{\mathbf{P}} = (V, E)$ where $V = S$ and $(s, s') \in E$ if and only if $\mathbf{P}(s, s') > 0$.

A *path* in \mathcal{D} is a (finite or infinite) sequence of states $\pi = s_0 s_1 s_2 \dots$ with $s_i \in S$ and $\mathbf{P}(s_i, s_{i+1}) > 0$ for all $i \geq 0$. The *probability of a finite path* $\hat{\pi} = s_0 \dots s_n$ is given by

$$\mathbf{P}(\hat{\pi}) := \prod_{i=0}^{n-1} \mathbf{P}(s_i, s_{i+1}).$$

For an atomic proposition $\mathbf{tar} \in AP$, we say that a state $s \in S$ is a *tar-state* if and only if $\mathbf{tar} \in L(s)$. We define $\Pi_s^{\diamond \mathbf{tar}}$ as the set of finite paths that start in s and end at the first visit of a *tar-state*, formally

$$\Pi_s^{\diamond \mathbf{tar}} := \{s_0 \dots s_n \mid s_0 \dots s_n \text{ is a finite path with } s_0 = s, \mathbf{tar} \in L(s_n) \text{ and } \mathbf{tar} \notin L(s_i) \text{ for } i < n\}.$$

We can assign rewards to states and transitions of a DTMC. Such an extended DTMC is called Markov reward model.

Definition 2.2 (Markov Reward Model)

A Markov reward model (MRM) is a tuple $\mathcal{M} = (\mathcal{D}, rew_{st}, rew_{tr})$ where

- $\mathcal{D} = (S, \mathbf{P}, \iota_{init}, AP, L)$ is the *underlying DTMC*,
- $rew_{st}: S \rightarrow \mathbb{R}_{\geq 0}$ is a *state reward function*, and
- $rew_{tr}: S \times S \rightarrow \mathbb{R}_{\geq 0}$ is a *transition reward function* with

$$\mathbf{P}(s, s') = 0 \text{ implies } rew_{tr}(s, s') = 0. \quad \blacksquare$$

In the following, let $\mathcal{M} = (\mathcal{D}, rew_{st}, rew_{tr})$ be an MRM with $\mathcal{D} = (S, \mathbf{P}, \iota_{init}, AP, L)$ and $s, s' \in S$.

The presented notions for DTMCs are also applicable for MRMs and refer to the underlying DTMC. For instance, we call \mathcal{M} *finite* if \mathcal{D} is finite and a path in \mathcal{M} refers to a path in \mathcal{D} .

Intuitively, the state reward $rew_{st}(s)$ is earned on leaving the state s , and the transition reward $rew_{tr}(s, s')$ is earned whenever the transition from s to s' has been taken. For a

finite path $\hat{\pi} = s_0 \dots s_n$ in \mathcal{M} , the (*cumulative*) *reward* of $\hat{\pi}$ is given by

$$\text{rew}(\hat{\pi}) := \sum_{i=0}^{n-1} (\text{rew}_{st}(s_i) + \text{rew}_{tr}(s_i, s_{i+1})).$$

Rewards can also be interpreted as costs without changing the definitions above. This results in a wide range of applications for MRMs. For instance, we can model power consumption of certain actions of a system by means of rewards in order to check properties regarding the expected power consumption of a system run. With the techniques described in the subsequent chapters, it is also possible to indicate why the expected power consumption is higher than a given threshold. In Chapter 1, we have seen another example where the rewards of an MRM are interpreted as attempts to send a message in a communication protocol. We will now give a slightly more complex example for such a protocol, where the rewards are interpreted as time.

Example 2.3 (MRM of a Communication Protocol)

Assume that a sender wants to send a message via an unreliable channel. Since the channel is used by multiple stations, it is possible that it is currently occupied. Before sending, the sender will check for 1 time unit, whether the channel is free. If this is not the case, it makes a probabilistic choice. With probability 0.25, the channel is checked again. With probability 0.75, the sender waits 4 time units and then, again, makes a probabilistic choice between the two alternatives. If the channel is free, the message will be sent. If the message has been transferred correctly, the sender receives an acknowledge from the receiver and the sending process ends successful. If there is a timeout (i.e., no acknowledge after 8 time units), the sender checks again whether the channel is free and will eventually make a new attempt to send the message. After three timeouts, the sending process ends unsuccessful. Providing that the channel worked properly, we assume that the acknowledge is received 3 time units after the sending attempt. At every moment, the channel is occupied with probability 0.1 and therefore free with probability 0.9. Then, the message or the acknowledge gets lost with probability 0.2. The channel works properly with probability 0.8.

Figure 2.1 on page 8 depicts an MRM \mathcal{M} that models the sender described in the communication protocol. The depiction is analogous to Figure 1.1 on page 3. Transition rewards are depicted as numbers on the corresponding transitions, after a vertical bar (i.e., a transition from s to s' is labeled with $\mathbf{P}(s, s') \mid \text{rew}_{tr}(s, s')$). The number is omitted if the reward is 0. The rewards in \mathcal{M} are interpreted as the number of time units required by the associated steps in the protocol.

The computation starts in the single initial state c_1 , where the channel is checked. If the sender detects that the channel is occupied, the system moves to o_1 , where the described probabilistic choice between waiting and checking is made. If the channel is free, the system moves to a_1 and attempts to send the message. The message is either successfully delivered (state *del*) or there is a timeout and the system moves to c_2 . The described

behavior is repeated and after the third timeout, the system moves to the absorbing state *err*.

In the following, we will see how the constructed model can be used to compute the probability for a delivery failure which corresponds to the probability of reaching a **failure-state**. Moreover, we will compute the expected time of a protocol run by considering the expected reward that is cumulated on a path from c_1 to a state labeled with **end**. ■

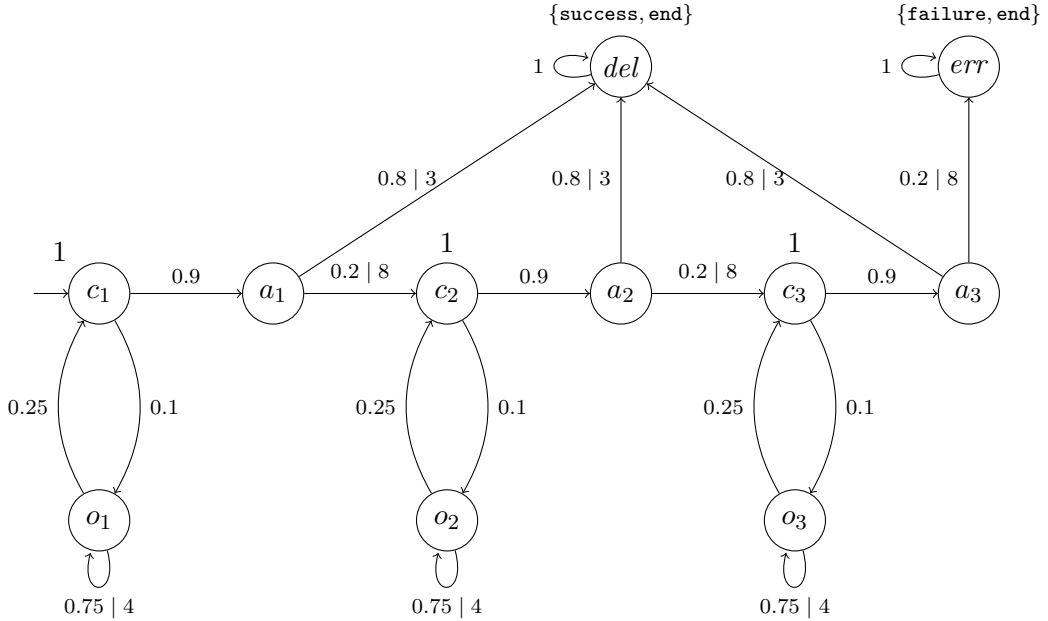


Figure 2.1: Markov reward model of the communication protocol described in Example 2.3.

2.2 Reachability Probabilities and Expected Rewards

There are several *properties* that can be checked for a DTMC or an MRM. The logic PRCTL is a common way to express a large part of these properties. However, only a small fraction of this logic is relevant for this work and therefore it is not explicitly introduced in this chapter. Instead of that, we are going to present the relevant parts and refer to [4] for more information on PRCTL.

First of all, we want to consider the following values regarding the probabilities of a DTMC and the rewards of an MRM:

- The *reachability probability*, i.e., the probability to reach a certain set of states when starting in a given state, and

- the *expected reward*, i.e., the expected amount of reward that has been cumulated until a certain set of states is reached when starting in a given state.

In the following, let $\mathcal{M} = (\mathcal{D}, rew_{st}, rew_{tr})$ be an MRM with the underlying DTMC $\mathcal{D} = (S, \mathbf{P}, \iota_{init}, AP, L)$.

Reachability probabilities. We briefly present the formal definition of reachability probabilities which makes use of cylinder sets of finite paths and requires some probability theory. A more detailed description can be found in [4].

A *cylinder set* $Cyl(\hat{\pi})$ of a finite path $\hat{\pi} = s_0 \dots s_n$ is the set of all infinite paths in \mathcal{D} that start with the sequence $s_0 \dots s_n$, i.e.,

$$Cyl(\hat{\pi}) := \{\pi \mid \pi \text{ is an infinite path with prefix } \hat{\pi}\}.$$

To formalize reachability probabilities, we associate \mathcal{D} with the smallest σ -algebra that contains the cylinder sets $Cyl(\hat{\pi})$ where $\hat{\pi}$ ranges over all finite paths in \mathcal{D} . The unique *probability measure* $Pr^{\mathcal{D}}$ of \mathcal{D} is defined by

$$Pr^{\mathcal{D}}(Cyl(s_0 \dots s_n)) := \iota_{init}(s_0) \cdot \mathbf{P}(s_0 \dots s_n).$$

Occasionally, we also write $Pr^{\mathcal{M}}$ instead of $Pr^{\mathcal{D}}$, where \mathcal{M} is an MRM with the underlying DTMC \mathcal{D} . If \mathcal{D} is clear from the context, the superscript can also be omitted and we simply write Pr . Let $\mathbf{tar} \in AP$ be an atomic proposition. The probability to reach a \mathbf{tar} -state when starting in a state $s \in S$ is given by

$$Pr^{\mathcal{D}}(s \models \diamond \mathbf{tar}) := \sum_{\hat{\pi} \in \Pi_s^{\diamond \mathbf{tar}}} Pr^{\mathcal{D}_s}(Cyl(\hat{\pi}))$$

where $\mathcal{D}_s = (S, \mathbf{P}, \iota_s, AP, L)$ is a DTMC derived from \mathcal{D} by replacing the initial distribution with

$$\iota_s(s') = \begin{cases} 1 & \text{if } s' = s \\ 0 & \text{otherwise.} \end{cases}$$

It should be noted that all paths $\hat{\pi} \in \Pi_s^{\diamond \mathbf{tar}}$ end at the first visit of a \mathbf{tar} state and therefore can not be a prefix of another path in this set. Hence, the sets $Cyl(\hat{\pi})$ for $\hat{\pi} \in \Pi_s^{\diamond \mathbf{tar}}$ are pairwise disjoint which justifies that the probabilities $Pr^{\mathcal{D}_s}(Cyl(\hat{\pi}))$ are summed up. If \mathcal{D} is finite, the reachability probabilities $p_s := Pr^{\mathcal{D}}(s \models \diamond \mathbf{tar})$ can be obtained for every state $s \in S$ by determining the sets

$$\begin{aligned} S_{=0} &:= \{s' \in S \mid \text{there is no } \mathbf{tar}\text{-state reachable from } s'\} \text{ and} \\ S_{=1} &:= \{s' \in S \mid \text{there is no state in } S_{=0} \text{ reachable from } s'\} \end{aligned}$$

and solving the linear equation system with the constraints

$$p_s = \begin{cases} 1 & \text{if } s \in S_{=1} \\ 0 & \text{if } s \in S_{=0} \\ \sum_{s' \in S} \mathbf{P}(s, s') \cdot p_{s'} & \text{otherwise} \end{cases}$$

for all $s \in S$. The set $S \setminus S_{=0}$ (i.e., the set of states from which a **tar**-state is reachable) can efficiently be computed by performing a backwards reachability analysis in the graph $\mathcal{G}_{\mathbf{P}}$ starting from all **tar**-states. Then, the set $S_{=0}$ can easily be obtained since it contains all states that are not in $S \setminus S_{=0}$. The set $S_{=1}$ can be determined in a similar way with the help of a backwards reachability analysis that starts in the states in $S_{=0}$.

Expected rewards. For the expected reward, we consider all paths that lead to a target state and weight the cumulative reward of a path with the probability of it. If $Pr(s \models \diamond \mathbf{tar}) = 1$ holds for an atomic proposition **tar** and a state s , then we can define the expected reward by

$$ExpRew^{\mathcal{M}}(s \models \diamond \mathbf{tar}) := \sum_{\hat{\pi} \in \Pi_s^{\diamond \mathbf{tar}}} Pr^{\mathcal{D}_s} (Cyl(\hat{\pi})) \cdot rew(\hat{\pi}).$$

Again, we omit the superscript \mathcal{M} , whenever the MRM is clear from the context. If $Pr(s \models \diamond \mathbf{tar}) < 1$ there is a positive probability that a **tar**-state will never be reached. Intuitively, an infinite amount of reward can be cumulated on such a system run. We therefore set $ExpRew(s \models \diamond \mathbf{tar}) := \infty$. Assuming that \mathcal{M} is finite and for all $s \in S$ it holds that $Pr(s \models \diamond \mathbf{tar}) = 1$, the expected rewards $r_s := ExpRew(s \models \diamond \mathbf{tar})$ are the unique solution of the linear equation system given by the constraints

$$r_s = \begin{cases} 0 & \text{if } \mathbf{tar} \in L(s) \\ rew_{st}(s) + \sum_{s' \in S} (\mathbf{P}(s, s') \cdot (rew_{tr}(s, s') + r_{s'})) & \text{otherwise} \end{cases}$$

for all $s \in S$.

State-properties. We can now define state-properties that can be satisfied or refuted in a given state.

Definition 2.4 (Reachability Property, Expected Reward Property)

Let $\mathcal{M} = (\mathcal{D}, rew_{st}, rew_{tr})$ be an MRM with $\mathcal{D} = (S, \mathbf{P}, \iota_{init}, AP, L)$, $s \in S$ and $\mathbf{tar} \in AP$. For a probability bound $\lambda \in [0, 1]$ and a comparison operator $\triangleleft \in \{<, \leq\}$, the reachability property $\mathbb{P}_{\triangleleft \lambda}(\diamond \mathbf{tar})$ is satisfied in s , written $s \models \mathbb{P}_{\triangleleft \lambda}(\diamond \mathbf{tar})$, if and only if $Pr(s \models \diamond \mathbf{tar}) \triangleleft \lambda$. For a reward bound $\lambda' \in \mathbb{R}_{\geq 0}$ and a comparison operator $\triangleleft \in \{<, \leq\}$, the expected reward property $\mathbb{E}_{\triangleleft \lambda'}(\diamond \mathbf{tar})$ is satisfied in s , written $s \models \mathbb{E}_{\triangleleft \lambda'}(\diamond \mathbf{tar})$, if and only if $ExpRew(s \models \diamond \mathbf{tar}) \triangleleft \lambda'$. ■

In the definition above, we only consider upper bound operators (either $<$ or \leq). A reachability property with a lower bound, i.e., a property of the form $\mathbb{P}_{\triangleright \lambda}(\diamond \mathbf{tar})$ with $\lambda \in [0, 1]$ and $\triangleright \in \{>, \geq\}$, can be converted to an equivalent reachability property with an upper bound. The conversion relies on the fact that we can consider the probability of the complementary event (“a **tar**-state will never be reached”). More information is given in [8]. For expected rewards, however, there is no complementary event and the

given conversion is not applicable. Possibly, there is no conversion from an expected reward property with a lower bound to an equivalent expected reward property with an upper bound. If this is the case, the results of this work are not applicable for properties of the form $\mathbb{E}_{\triangleright\lambda}(\diamond\mathbf{tar})$ with $\lambda \in \mathbb{R}_{\geq 0}$ and $\triangleright \in \{>, \geq\}$.

For the rest of this work, whenever a state property $\mathbb{P}_{\triangleleft\lambda}(\diamond\mathbf{tar})$ (or $\mathbb{E}_{\triangleleft\lambda'}(\diamond\mathbf{tar})$) is considered, we assume that \mathbf{tar} is an atomic proposition, $\lambda \in [0, 1]$ (or $\lambda' \in \mathbb{R}_{\geq 0}$) is a probability (or reward) bound and \triangleleft is an upper bound operator (either $<$ or \leq). The states that are labeled with the atomic proposition \mathbf{tar} are also called *target states* (with respect to the property $\mathbb{P}_{\triangleleft\lambda}(\diamond\mathbf{tar})$ or $\mathbb{E}_{\triangleleft\lambda'}(\diamond\mathbf{tar})$). If a property is refuted in a state s , we write $s \not\models \mathbb{P}_{\triangleleft\lambda}(\diamond\mathbf{tar})$ (or $s \not\models \mathbb{E}_{\triangleleft\lambda'}(\diamond\mathbf{tar})$). Finally, a state $s \in S$ is called *relevant* for $Pr(s_{init} \models \diamond\mathbf{tar})$ (or $ExpRew(s_{init} \models \diamond\mathbf{tar})$) if there is a path $\hat{\pi} \in \Pi_{s_{init}}^{\diamond\mathbf{tar}}$ that visits s . Intuitively, the relevant states are exactly the states that “contribute” to the considered reachability probability or the expected reward. The transition probabilities and the rewards of the remaining states do not affect this values.

Example 2.5 (Properties for a Communication Protocol)

Consider the communication protocol described in Example 2.3 on page 7 and the corresponding MRM $\mathcal{M} = (\mathcal{D}, rew_{st}, rew_{tr})$, depicted in Figure 2.1 on page 8. Assume that there are two requirements:

1. The sending process ends erroneous with probability less than 0.006.
2. The expected time required by the sending process is less than 7 time units.

The probability for an erroneous end of the sending process agrees with the probability of reaching a **failure**-state, starting from the state c_1 . Thus, the first requirement holds if and only if

$$c_1 \models \mathbb{P}_{<0.006}(\diamond\mathbf{failure}).$$

To check whether this property holds, we determine $Pr(c_1 \models \diamond\mathbf{failure})$ by solving a linear equation system as it is described above. It is easy to see that $S_{=0} = \{del\}$ and $S_{=1} = \{err\}$. Let $p_s = Pr(s \models \diamond\mathbf{failure})$ for all states s of the depicted model. It holds that $p_{del} = 0$ and $p_{err} = 1$. The remaining probabilities can be calculated as follows:

$$\begin{array}{ll} p_{a_3} = 0.8 \cdot p_{del} + 0.2 \cdot p_{err} & \Rightarrow p_{a_3} = 0.2 \\ p_{o_3} = 0.75 \cdot p_{o_3} + 0.25 \cdot p_{c_3} & \Rightarrow p_{o_3} = p_{c_3} = 0.2 \\ p_{c_3} = 0.1 \cdot p_{o_3} + 0.9 \cdot p_{a_3} & \Rightarrow p_{c_3} = 0.2 \\ p_{a_2} = 0.8 \cdot p_{del} + 0.2 \cdot p_{c_3} & \Rightarrow p_{a_2} = 0.04 \\ p_{o_2} = 0.75 \cdot p_{o_2} + 0.25 \cdot p_{c_2} & \Rightarrow p_{o_2} = p_{c_2} = 0.04 \\ p_{c_2} = 0.1 \cdot p_{o_2} + 0.9 \cdot p_{a_2} & \Rightarrow p_{c_2} = 0.04 \\ p_{a_1} = 0.8 \cdot p_{del} + 0.2 \cdot p_{c_2} & \Rightarrow p_{a_1} = 0.008 \\ p_{o_1} = 0.75 \cdot p_{o_1} + 0.25 \cdot p_{c_1} & \Rightarrow p_{o_1} = p_{c_1} = 0.008 \\ p_{c_1} = 0.1 \cdot p_{o_1} + 0.9 \cdot p_{a_1} & \Rightarrow p_{c_1} = 0.008 \end{array}$$

We can conclude that the first requirement does not hold since

$$p_{c_1} = Pr(c_1 \models \diamond \text{failure}) = 0.008 \not\leq 0.006.$$

The expected time required by the sending process is given by the expected reward that has been cumulated until an **end**-state is reached, starting from the state c_1 . The second requirement of our protocol holds if and only if

$$c_1 \models \mathbb{E}_{<7}(\diamond \text{end}).$$

We solve another equation system in order to obtain the value $ExpRew(c_1 \models \diamond \text{end})$. Let $r_s = ExpRew(s \models \diamond \text{end})$ for all states s of the depicted model. We get $r_{del} = r_{err} = 0$, as **end** $\in L(del)$ and **end** $\in L(err)$. The remaining expected rewards can be calculated as follows:

$$\begin{aligned} r_{a_3} &= 0.8 \cdot (r_{del} + 3) + 0.2 \cdot (r_{err} + 8) && \Rightarrow r_{a_3} = 4 \\ r_{o_3} &= 0.75 \cdot (r_{o_3} + 4) + 0.25 \cdot r_{c_3} && \Rightarrow r_{o_3} = 12 + r_{c_3} = 18.444 \\ r_{c_3} &= 1 + 0.1 \cdot r_{o_3} + 0.9 \cdot r_{a_3} && \Rightarrow r_{c_3} = 6.444 \\ r_{a_2} &= 0.8 \cdot (r_{del} + 3) + 0.2 \cdot (r_{c_3} + 8) && \Rightarrow r_{a_2} = 5.288 \\ r_{o_2} &= 0.75 \cdot (r_{o_2} + 4) + 0.25 \cdot r_{c_2} && \Rightarrow r_{o_2} = 12 + r_{c_2} = 19.733 \\ r_{c_2} &= 1 + 0.1 \cdot r_{o_2} + 0.9 \cdot r_{a_2} && \Rightarrow r_{c_2} = 7.733 \\ r_{a_1} &= 0.8 \cdot (r_{del} + 3) + 0.2 \cdot (r_{c_2} + 8) && \Rightarrow r_{a_1} = 5.547 \\ r_{o_1} &= 0.75 \cdot (r_{o_1} + 4) + 0.25 \cdot r_{c_1} && \Rightarrow r_{o_1} = 12 + r_{c_1} = 19.991 \\ r_{c_1} &= 1 + 0.1 \cdot r_{o_1} + 0.9 \cdot r_{a_1} && \Rightarrow r_{c_1} = 7.991 \end{aligned}$$

the second requirement is refuted as well, since

$$r_{c_1} = ExpRew(c_1 \models \diamond \text{end}) = 7.991 \not\leq 7, \quad \blacksquare$$

2.3 Restrictions

For the rest of this work, we restrict ourselves to finite DTMCs and finite MRMs. Thus, reachability probabilities and expected rewards can efficiently be computed by solving linear equation systems.

Furthermore, we only consider *MRMs without transition rewards*. An MRM $\mathcal{M} = (\mathcal{D}, rew_{st}, rew_{tr})$ has no transition rewards if $rew_{tr}(s, s') = 0$ for all states s and s' . If this is the case, we also write $\mathcal{M} = (\mathcal{D}, rew_{st})$ or simply $\mathcal{M} = (\mathcal{D}, rew)$. If a finite MRM $\mathcal{M} = (\mathcal{D}, rew_{st}, rew_{tr})$ with $\mathcal{D} = (S, \mathbf{P}, \iota_{init}, AP, L)$ has transition rewards, we can construct an MRM $\mathcal{M}' = (\mathcal{D}', rew')$ without transition rewards such that the states

in \mathcal{M} and \mathcal{M}' satisfy the same reachability and expected reward properties. This can be done by setting $\mathcal{D}' = \mathcal{D}$ and defining the state rewards:

$$rew'(s) = rew_{st}(s) + \sum_{s' \in S} \mathbf{P}(s, s') \cdot rew_{tr}(s, s').$$

With the constructed MRM $\mathcal{M}' = (\mathcal{D}', rew')$, for all states s and atomic propositions \mathbf{tar} it holds that

$$\begin{aligned} Pr^{\mathcal{D}}(s \models \diamond \mathbf{tar}) &= Pr^{\mathcal{D}'}(s \models \diamond \mathbf{tar}) \text{ and} \\ ExpRew^{\mathcal{M}}(s \models \diamond \mathbf{tar}) &= ExpRew^{\mathcal{M}'}(s \models \diamond \mathbf{tar}). \end{aligned}$$

The first statement is trivial, since the underlying DTMC is not affected. If $Pr(s \models \diamond \mathbf{tar}) < 1$, we get $ExpRew^{\mathcal{M}}(s \models \diamond \mathbf{tar}) = \infty = ExpRew^{\mathcal{M}'}(s \models \diamond \mathbf{tar})$. Otherwise, the second statement can be proven by considering the equation system characterization of expected rewards.

For simplicity, we assume that every target state is absorbing, i.e., $\mathbf{P}(s, s) = 1$ and $\mathbf{P}(s, s') = 0$ for all \mathbf{tar} -states s and states $s' \neq s$. We also assume that a DTMC $\mathcal{D} = (S, \mathbf{P}, \nu_{init}, AP, L)$ only has a *single initial state*. This is the case if there is a state $s_{init} \in S$ such that

$$\nu_{init}(s) = \begin{cases} 1 & \text{if } s = s_{init} \\ 0 & \text{otherwise.} \end{cases}$$

In this case, we simply write $\mathcal{D} = (S, \mathbf{P}, s_{init}, AP, L)$. Occasionally, we also write $\mathcal{D} \models \mathbb{P}_{\leq \lambda}(\diamond \mathbf{tar})$ instead of $s_{init} \models \mathbb{P}_{\leq \lambda}(\diamond \mathbf{tar})$. This also applies for expected reward properties as well as for MRMs where the underlying DTMC has a single initial state.

Consider an MRM $\mathcal{M} = (\mathcal{D}, rew)$ with a DTMC $\mathcal{D} = (S, \mathbf{P}, \nu_{init}, AP, L)$ and $\mathbf{tar} \in AP$. We construct the MRM $\mathcal{M}' = (\mathcal{D}', rew')$ with the underlying DTMC $\mathcal{D}' = (S \uplus \{s_{init}\}, \mathbf{P}', s_{init}, AP, L')$ that has a single initial state s_{init} and all \mathbf{tar} -states are absorbing. For this purpose, we define the outgoing transitions of the new state s_{init} according to the initial distribution ν_{init} and redirect the outgoing transitions of \mathbf{tar} states to the state itself. Let $T = \{s \in S \mid \mathbf{tar} \in L(s)\}$ be the set of all \mathbf{tar} -states. We set:

$$\begin{aligned} \bullet \mathbf{P}'(s, s') &= \begin{cases} \nu_{init}(s') & \text{if } s = s_{init} \text{ and } s' \in S \\ \mathbf{P}(s, s') & \text{if } s \in (S \setminus T) \text{ and } s' \in S \\ 1 & \text{if } s \in T \text{ and } s' = s \\ 0 & \text{otherwise,} \end{cases} \\ \bullet L'(s) &= \begin{cases} \emptyset & \text{if } s = s_{init} \\ L(s) & \text{otherwise,} \end{cases} \\ &\text{and} \\ \bullet rew'(s) &= \begin{cases} 0 & \text{if } s = s_{init} \\ rew(s) & \text{otherwise.} \end{cases} \end{aligned}$$

Note that the new state s_{init} can not be reached from any other state. Furthermore, for all states s there is no path in $\Pi_s^{\diamond \mathbf{tar}}$ which uses outgoing transitions of \mathbf{tar} -states. Hence, the construction has no effect on the reachability and expected reward properties, i.e., for all $s \in S$, it holds that

$$\begin{aligned} Pr^{\mathcal{D}}(s \models \diamond \mathbf{tar}) &= Pr^{\mathcal{D}'}(s \models \diamond \mathbf{tar}) \text{ and} \\ ExpRew^{\mathcal{M}}(s \models \diamond \mathbf{tar}) &= ExpRew^{\mathcal{M}'}(s \models \diamond \mathbf{tar}). \end{aligned}$$

Furthermore, the reachability probability as well as the expected reward of s_{init} corresponds to the values of all states of the original system weighted with the probabilities given by the initial distribution ι_{init} , i.e.,

$$\begin{aligned} Pr^{\mathcal{D}'}(s_{init} \models \diamond \mathbf{tar}) &= \sum_{s \in S} \iota_{init}(s) \cdot Pr^{\mathcal{D}}(s \models \diamond \mathbf{tar}) \text{ and} \\ ExpRew^{\mathcal{M}'}(s_{init} \models \diamond \mathbf{tar}) &= \sum_{s \in S} \iota_{init}(s) \cdot ExpRew^{\mathcal{M}}(s \models \diamond \mathbf{tar}). \end{aligned}$$

Example 2.6 (Constructing an MRM that meets the Restrictions)

Consider again the MRM \mathcal{M} of the communication protocol from Example 2.3, depicted in Figure 2.1 on page 8 as well as the property $\mathbb{E}_{<7}(\diamond \mathbf{end})$ from Example 2.5 on page 11. Applying the constructions above yields an MRM \mathcal{M}' without transition rewards, a single initial state as well as absorbing target states. The resulting MRM is depicted in Figure 2.2. It holds that:

$$\underbrace{ExpRew(c_1 \models \diamond \mathbf{end})}_{for \mathcal{M}} = \underbrace{ExpRew(s_{init} \models \diamond \mathbf{end})}_{for \mathcal{M}'} = 7.991. \quad \blacksquare$$

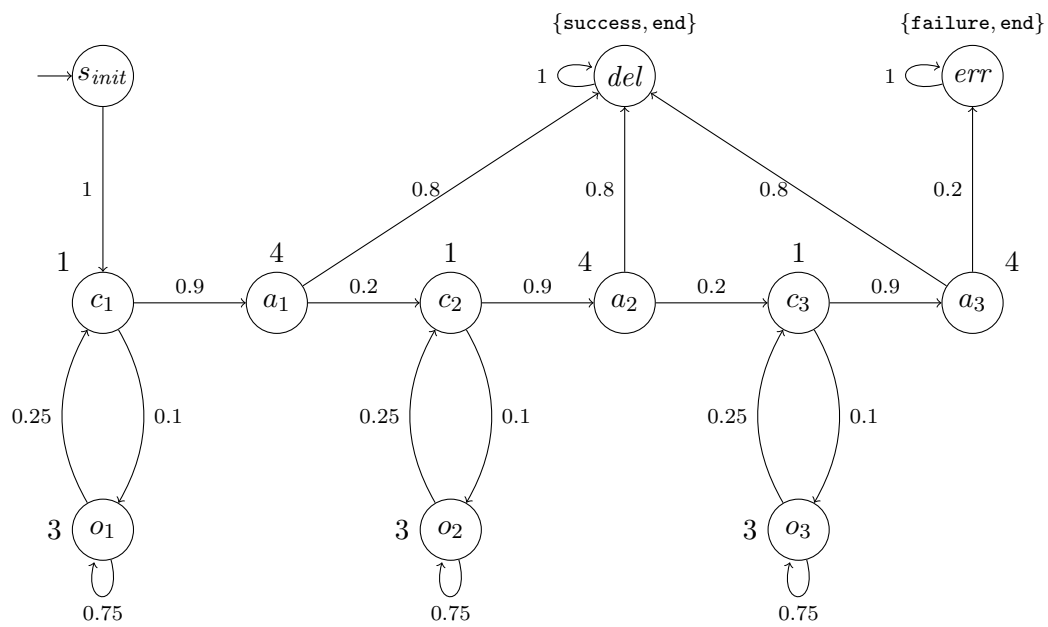


Figure 2.2: Markov reward model of the communication protocol without transition rewards, with a single initial state, and absorbing target states (with respect to the property $\mathbb{E}_{<7}(\diamond \text{end})$).

Chapter 3

Related Work

In this chapter, let $\mathcal{D} = (S, \mathbf{P}, s_{init}, AP, L)$ be a finite DTMC. There are several approaches to demonstrate why a reachability property is refuted in the initial state s_{init} of \mathcal{D} . An overview of these approaches is provided in [1]. We want to take a closer look at counterexamples based on path enumeration and the idea of critical subsystems.

3.1 Path-based Counterexamples for Reachability Properties

In [8], a counterexample for a statement of the form $s_{init} \models \mathbb{P}_{\triangleleft\lambda}(\diamond\mathbf{tar})$ is defined as a set of finite paths, starting in s_{init} and leading to a \mathbf{tar} -state, such that enough probability mass is induced.

Definition 3.1 (Counterexample for Reachability Properties)

Let $\mathcal{D} = (S, \mathbf{P}, s_{init}, AP, L)$ be a DTMC and $\mathbb{P}_{\triangleleft\lambda}(\diamond\mathbf{tar})$ a reachability property. If the property is refuted in the initial state s_{init} , a *counterexample* for $\mathbb{P}_{\triangleleft\lambda}(\diamond\mathbf{tar})$ is a set of paths $\Pi_{CEX} \subseteq \Pi_{s_{init}}^{\diamond\mathbf{tar}}$ such that $\mathbf{P}(\Pi_{CEX}) \triangleright \lambda$, where

$$\mathbf{P}(\Pi_{CEX}) := \sum_{\hat{\pi} \in \Pi_{CEX}} \mathbf{P}(\hat{\pi}) \quad \text{and} \quad \triangleright := \begin{cases} \geq & \text{if } \triangleleft = < \\ > & \text{if } \triangleleft = \geq. \end{cases} \quad \blacksquare$$

Due to $\Pi_{CEX} \subseteq \Pi_{s_{init}}^{\diamond\mathbf{tar}}$, there can not be a path in Π_{CEX} that is a prefix of another path in this set. Therefore, the probabilities of the considered paths can be summed up. Since

$$\mathbf{P}(\Pi_{s_{init}}^{\diamond\mathbf{tar}}) = Pr(s_{init} \models \diamond\mathbf{tar}),$$

there is always a counterexample given by $\Pi_{CEX} = \Pi_{s_{init}}^{\diamond\mathbf{tar}}$.

Π_{CEX} is called a minimal counterexample if $|\Pi_{CEX}| \leq |\Pi'_{CEX}|$ for all counterexamples Π'_{CEX} . Π_{CEX} is a smallest counterexample if it is minimal and $\mathbf{P}(\Pi_{CEX}) \geq \mathbf{P}(\Pi'_{CEX})$ for all minimal counterexamples Π'_{CEX} . The computation of a smallest counterexample can be accomplished by solving an instance of the k -shortest path problem. The number of required paths (k) is determined on the fly by repeatedly finding a shortest, not yet considered path until the set of found paths induces enough probability mass. More information on this method can be found in [8].

Example 3.2 (Counterexample for a DTMC represented by a set of paths)

Consider the DTMC depicted in Figure 3.1 and the property $\mathbb{P}_{<0.5}(\diamond \text{tar})$ which is violated in the initial state s_1 . There is only one counterexample given by the infinite set $\Pi_{s_1}^{\diamond \text{tar}} = \{(s_1)^n s_2 \mid n > 0\}$. Since there is no other counterexample, $\Pi_{s_1}^{\diamond \text{tar}}$ is minimal and a smallest counterexample. ■

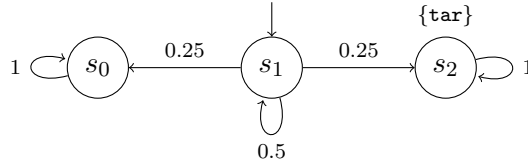


Figure 3.1: DTMC with no finite counterexample for $s_1 \models \mathbb{P}_{<0.5}(\diamond \text{tar})$.

3.2 Critical Subsystems for Reachability Properties

In Example 3.2, we have seen that smallest counterexamples represented by sets of paths can be infinite. Even if there is a finite counterexample, it often consists of a large set of paths which leads to high memory consumption and is not useful for practical applications. An alternative approach is to generate a critical subsystem [2, 11]. The idea is to select states of the original system in order to form a subsystem that already refutes the property.

Definition 3.3 (Selection, Critical Subsystem for a Reachability Property)

Let $\mathcal{D} = (S, \mathbf{P}, s_{init}, AP, L)$ be a DTMC. A *selection* is a set of states S' such that $s_{init} \in S' \subseteq S$. The *subsystem* of \mathcal{D} induced by a selection $S' \subseteq S$ is given by a DTMC $\mathcal{D}' = (S' \uplus \{s_{\perp}\}, \mathbf{P}', s_{init}, AP, L')$ where $s_{\perp} \notin S$ is a new state and

$$\bullet \mathbf{P}'(s, s') = \begin{cases} \mathbf{P}(s, s') & \text{for } s, s' \in S' \\ \sum_{s'' \in S \setminus S'} \mathbf{P}(s, s'') & \text{for } s \in S' \text{ and } s' = s_{\perp} \\ 1 & \text{for } s = s' = s_{\perp} \\ 0 & \text{for } s = s_{\perp} \text{ and } s' \in S' \end{cases}$$

and

$$\bullet L'(s) = \begin{cases} L(s) & \text{for } s \in S' \\ \emptyset & \text{for } s = s_{\perp}. \end{cases}$$

The subsystem \mathcal{D}' is *critical* for a property $\mathbb{P}_{\triangleleft\lambda}(\diamond\mathbf{tar})$ if $\mathcal{D}' \not\models \mathbb{P}_{\triangleleft\lambda}(\diamond\mathbf{tar})$. ■

We are aimed to find critical subsystems induced by a preferably small set of selected states. Ideally, the considered selection $S' \subseteq S$ is minimal, i.e., if a selection S'' induces a critical subsystem, then $|S'| \leq |S''|$ holds. In [19], Wimmer et al. introduced techniques to compute such minimal critical subsystems. However, these techniques exhibit high time consumption and might therefore be impractical for large inputs. In [13] and [14], the authors present the generation of (not necessarily minimal) critical subsystems using symbolic representations, meaning that (multi-terminal) binary decision diagrams are used to store the data. The symbolic approach leads to fast computations even for large inputs since certain operations can be performed very efficient.

In this thesis, however, we assume that DTMCs (and MRMs) are given in an explicit representation where sparse matrices are used to store the transition probabilities. The treatment of critical subsystems for expected rewards using symbolic representations is left for future work. Furthermore, we focus on heuristic approaches to efficiently generate small critical subsystems of DTMCs (and later MRMs). These approaches provide a balance between fast computation and a small set of selected states. We will now briefly present the *extended best-first search* [2] and the *local path search* [11]. More information on this methods is provided by the cited works.

Extended best-first search. Given a DTMC $\mathcal{D} = (S, \mathbf{P}, s_{init}, AP, L)$ and a reachability property $\mathbb{P}_{\triangleleft\lambda}(\diamond\mathbf{tar})$ refuted in s_{init} , the extended best-first search generates a selection $S' \subseteq S$ that induces a critical subsystem as follows:

There are two lists of states, namely **open** and **closed**. The **open** list is sorted according to a function $f: S \rightarrow \mathbb{R}$. This function evaluates how beneficial it is to add a given state to the selection S' . We will formally define f later. Initially, we have $S' := \{s_{init}\}$ and s_{init} is inserted into **open** while **closed** remains empty. After that, repeatedly a state s with $f(s) = \max\{f(s') \mid s' \in \mathbf{open}\}$ is moved from **open** to **closed** and the states s' with $\mathbf{P}(s, s') > 0$, $\mathbf{tar} \notin L(s')$ and $s' \notin \mathbf{closed}$ are inserted into **open**. For all states in the two lists, except s_{init} , the incoming transitions that lead to that state are stored. Whenever a **tar**-state or a state that is already contained in S' is found, this information is used to extend the selection S' with the **tar**-state (if applicable) as well as the states that are backwards reachable from the currently processed state. The procedure terminates, if S' induces a critical subsystem.

The function f is defined as $f(s) = g(s) \cdot h(s)$, with $g, h: S \rightarrow \mathbb{R}$. The idea is that the values $g(s)$ and $h(s)$ approximate the probability to reach the state s from s_{init} and to reach a **tar**-state from s , respectively. While the definition of $h(s)$ depends on the particular application, $g(s)$ is defined as the probability of the most probable path

from s_{init} to s that has been found so far. This is achieved by setting $g(s_{init}) = 1$ and updating the value $g(s')$ to $g(s) \cdot \mathbf{P}(s, s')$ whenever a state s' is found while processing a state s , such that either $g(s')$ is not yet defined or the old value of $g(s')$ is less than $g(s) \cdot \mathbf{P}(s, s')$. If $s' \in \mathbf{closed}$ and the value $g(s')$ is updated to a larger value, s' is moved from **closed** to **open** to take the higher value of f into consideration for the succeeding computation.

Note that the original DTMC \mathcal{D} is explored on the fly. This is a big advantage if the state space is very large and the model has not been build yet since only the parts that are relevant for the search have to be generated. On the other hand, it is not easy to obtain an accurate definition for the function h without exploring the model in advance. This can lead to a badly informed search that explores unpromising parts of the model.

Example 3.4 (Extended Best-first Search for Communication Protocol)

Let $\mathcal{D} = (S, \mathbf{P}, c_1, AP, L)$ be the DTMC of the communication protocol from Example 2.3 on page 2.3, again depicted at the top of Figure 3.2 (we only consider the underlying DTMC of the constructed MRM). Moreover, consider the property $\mathbb{P}_{<0.006}(\diamond \mathbf{failure})$ from Example 2.5 on page 11. We already know that this property is refuted in the initial state c_1 . We use the extended best first search to generate a critical subsystem of \mathcal{D} . For simplicity, we are going to assume that $h(s) = 1$ for all states $s \in S$ and therefore $f(s) = g(s)$. The selection S' and the sorted list **open** initially contain the state c_1 . In the first step, c_1 is moved from **open** to **closed**. The direct successors of c_1 , namely a_1 and o_1 , are inserted into **open**. a_1 is inserted before o_1 , since $f(a_1) = 0.9 > 0.1 = f(o_1)$. In the second step, a_1 is moved from **open** to **closed** and its successor are added to **open**. The procedure goes on as follows:

open = (c_1)	closed = ()	$f(c_1) = 1$
open = (a_1, o_1)	closed = (c_1)	$f(a_1) = 0.9, f(o_1) = 0.1$
open = (del, c_2, o_1)	closed = (c_1, a_1)	$f(del) = 0.27, f(c_2) = 0.18$
open = (c_2, o_1)	closed = (c_1, a_1, del)	
open = (a_2, o_1, o_2)	closed = (c_1, a_1, del, c_2)	$f(a_2) = 0.144, f(o_2) = 0.018$
open = (o_1, c_3, o_2)	closed = (c_1, a_1, del, c_2, a_2)	$f(c_3) = 0.0288$

While exploring the successors of o_1 , the state $c_1 \in S'$ has been found and the selection is updated to $S' = \{c_1, o_1\}$. For the subsystem \mathcal{D}' of \mathcal{D} induced by S' , it is obvious that $Pr^{\mathcal{D}'}(c_1 \models \diamond \mathbf{failure}) = 0$ since it contains no **failure**-state. Therefore \mathcal{D}' is not critical yet and the search is continued:

open = (c_3, o_2)	closed = ($c_1, a_1, del, c_2, a_2, o_1$)	
open = (a_3, o_2, o_3)	closed = ($c_1, a_1, del, c_2, a_2, o_1, c_3$)	$f(a_3) = 0.02592$ $f(o_3) = 0.00288$

While exploring the successors of a_3 , the **failure**-state err has been found. Moreover, the states c_3, a_2, c_2 and a_1 are backwards reachable. Hence, the selection is updated to $S' = \{c_1, o_1, a_1, c_2, a_2, c_3, a_3, err\}$. The subsystem \mathcal{D}' of \mathcal{D} induced by S' is depicted at the bottom of Figure 3.2. It holds that $Pr^{\mathcal{D}'}(c_1 \models \diamond \mathbf{failure}) = 0.00648 \geq 0.006$.

Therefore, the subsystem \mathcal{D}' is critical, i.e., $c_1 \not\models \mathbb{P}_{<0.006}(\diamond \text{failure})$. ■

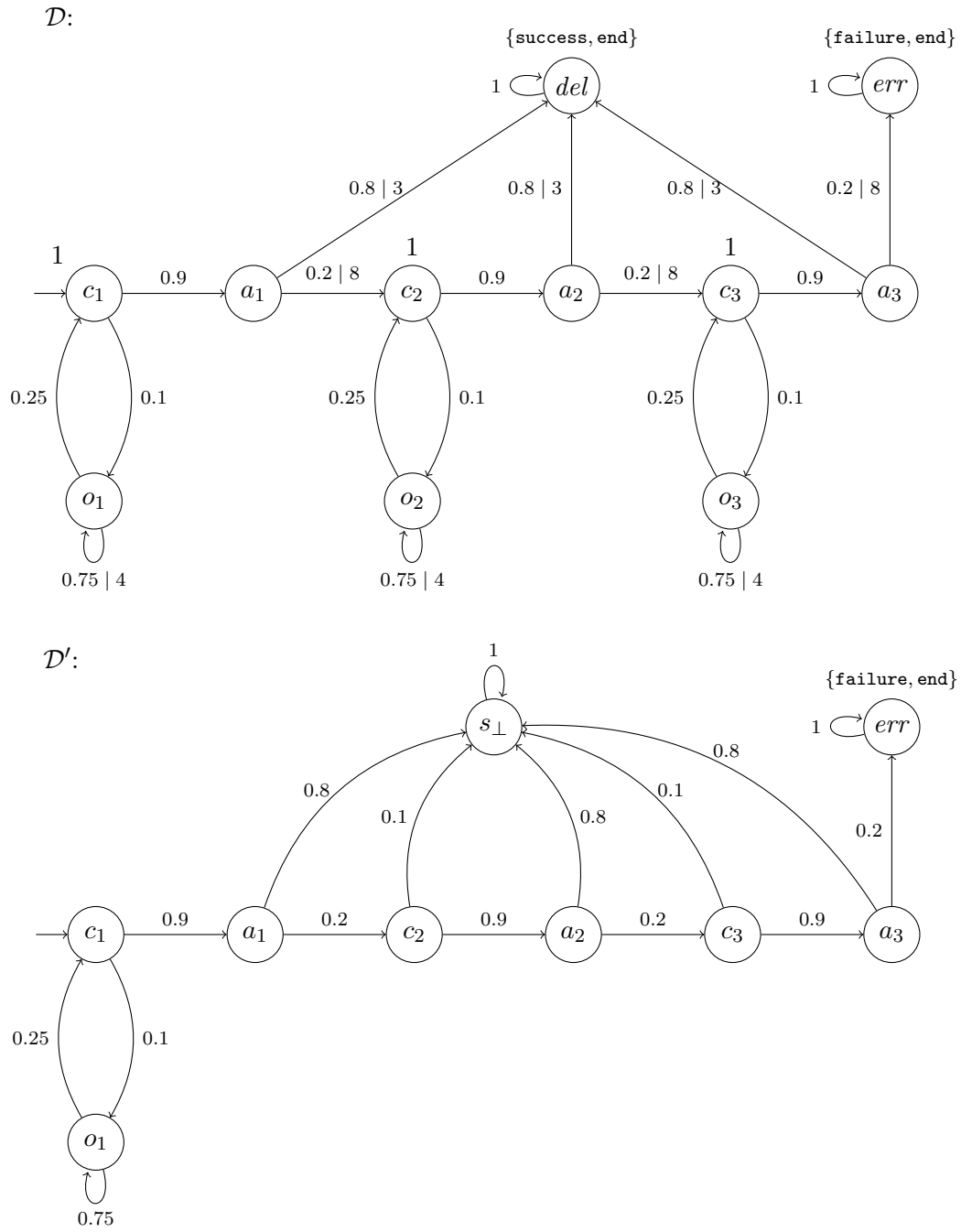


Figure 3.2: Original DTMC \mathcal{D} (top) and the critical subsystem \mathcal{D}' of \mathcal{D} generated by the extended best first search approach (bottom).

Local path search. Given a DTMC $\mathcal{D} = (S, \mathbf{P}, s_{init}, AP, L)$ and a reachability property $\mathbb{P}_{<\lambda}(\diamond \mathbf{tar})$ refuted in s_{init} , the local path search finds paths that connect already selected states with target states or, again, selected states. A selection $S' \subseteq S$ is extended by the states visited on these paths until it induces a critical subsystem.

Let $\Delta_{S_{start}}^{S_{end}}$ denote the set of finite paths in \mathcal{D} that connect two sets of states $S_{start} \subseteq S$ and $S_{end} \subseteq S$. Paths that intermediately visit $S_{start} \cup S_{end}$ or never leave S_{start} are excluded. Formally:

$$\Delta_{S_{start}}^{S_{end}} := \{s_0 \dots s_n \mid s_0 \dots s_n \text{ is a finite path with } s_0 \in S_{start}, s_n \in S_{end}, \\ s_i \notin (S_{start} \cup S_{end}) \text{ for } 0 < i < n \text{ and } (n > 1 \text{ or } s_n \notin S_{start})\}.$$

Note that S_{start} and S_{end} do not have to be disjoint. A most probable path connecting S_{start} with S_{end} is a path $\hat{\pi} \in \Delta_{S_{start}}^{S_{end}}$ such that

$$\mathbf{P}(\hat{\pi}) \geq \mathbf{P}(\hat{\pi}') \text{ for all } \hat{\pi}' \in \Delta_{S_{start}}^{S_{end}}.$$

Let T be the set of all **tar**-states. The first step of the local path search is to find a most probable path connecting $\{s_{init}\}$ with T . The selection $S' \subseteq S$ is initialized with the states that are visited on the found path. After that, a most probable path that connects S' with $S' \cup T$ is searched and the selection S' is extended by the states visited on that path. This step is repeated until the selection S' induces a critical subsystem. Finding the required paths can be done with a modified version of Dijkstra's shortest path algorithm. A more detailed description of this procedure can also be found in Chapter 5.

Example 3.5 (Local Path Search for Communication Protocol)

Again, let $\mathcal{D} = (S, \mathbf{P}, c_1, AP, L)$ be the DTMC of the communication protocol from Example 2.3, depicted on the top of Figure 3.2 on page 21. Let us now apply the local path search to generate a subsystem of \mathcal{D} that is critical for the property $\mathbb{P}_{<0.006}(\diamond \mathbf{failure})$ from Example 2.5 on page 11. At first, a most probable path between $\{c_1\}$ and the set of **failure**-states (i.e., $\{err\}$) is searched. This path is unique and given by:

$$\pi_0 := c_1 a_1 c_2 a_2 c_3 a_3 err.$$

The selection $S' \subseteq S$ is initialized by the states visited on π_0 , i.e., $S' = \{c_1, a_1, c_2, a_2, c_3, a_3, err\}$. The subsystem \mathcal{D}_0 of \mathcal{D} induced by S' is depicted in Figure 3.3 (top). In this DTMC, it holds that

$$Pr^{\mathcal{D}_0}(c_1 \models \diamond \mathbf{failure}) = 0.005832 < 0.006.$$

Hence, the subsystem is not critical yet. The next step is to find a most probable path connecting S' with $S' \cup \{err\}$. There are three possibilities:

$$\pi_1^{(i)} := c_i o_i c_i \quad \text{for } i \in \{1, 2, 3\}.$$

Extending the selection S' with the states visited on $\pi_1^{(1)}$ yields $S' = \{c_1, a_1, o_1, c_2, a_2, c_3, a_3, err\}$. The subsystem \mathcal{D}_1 of \mathcal{D} induced by the updated selection S' is depicted in Figure 3.3 (bottom). Now it holds

$$Pr^{\mathcal{D}_1}(c_1 \models \diamond failure) = 0.00648 \geq 0.006.$$

Therefore, the subsystem \mathcal{D}_1 is critical, i.e., $c_1 \not\models \mathbb{P}_{<0.006}(\diamond failure)$. ■

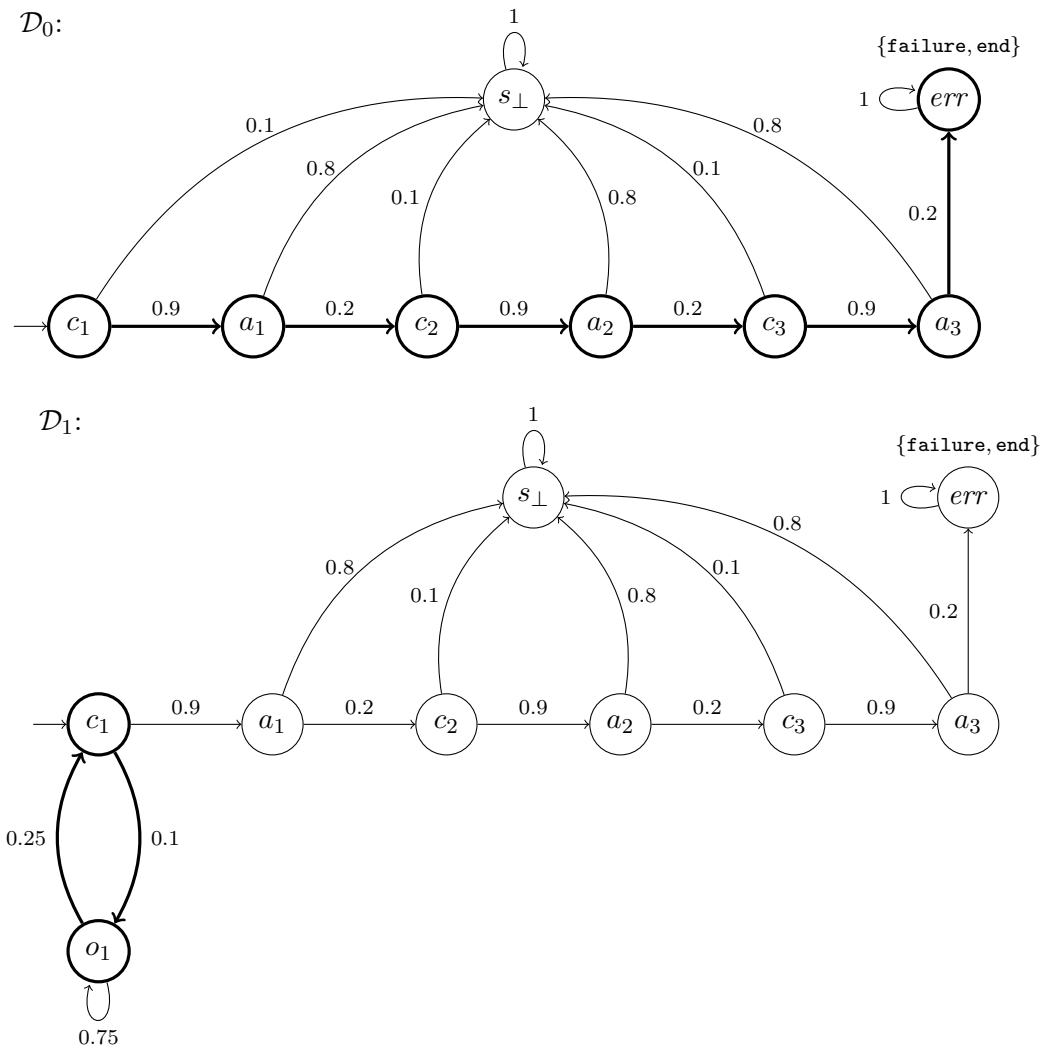


Figure 3.3: Subsystems \mathcal{D}_0 for the path π_0 and \mathcal{D}_1 for the paths $\pi_0, \pi_1^{(1)}$. The transitions and states along the recently found path are marked with thick lines.

3.3 Tools and Implementations

PRISM [17] and MRMC [15] are famous tools where model checking algorithms for the models and properties described in Chapter 2 have been implemented. PRISM also introduces a human readable high-level language to describe DTMCs, MRMs and other types of probabilistic models. These models can be exported to an explicit representation which can be used as input for other tools like MRMC.

The tools DiPro [3] and COMICS [12] are dedicated for the generation of counterexamples. The extended best-first search presented above has been implemented in DiPro. An implementation of the local path search is provided by the tool COMICS [12].

Chapter 4

Critical Subsystems Applied to Expected Reward Properties

In the previous chapter, critical subsystems for reachability properties have been presented. We will now discuss how this notion can be applied for expected reward properties.

In the following, consider a finite MRM $\mathcal{M} = (\mathcal{D}, rew)$ without transition rewards and with the underlying DTMC $\mathcal{D} = (S, \mathbf{P}, s_{init}, AP, L)$. Moreover, let $\mathbb{E}_{\triangleleft\lambda}(\diamond\mathbf{tar})$ be an expected reward property such that $\mathcal{M} \not\models \mathbb{E}_{\triangleleft\lambda}(\diamond\mathbf{tar})$.

4.1 Reachability of Target States

Before we formally define critical subsystems for expected reward properties, we discuss how the target states of \mathcal{M} can be reached.

At first, let us assume that $Pr(s_{init} \models \diamond\mathbf{tar}) < 1$. From the definition of expected rewards, we can derive $ExpRew(s_{init} \models \diamond\mathbf{tar}) = \infty$ and therefore $s_{init} \not\models \mathbb{E}_{\triangleleft\lambda}(\diamond\mathbf{tar})$ for arbitrary \triangleleft and λ . Hence, it is sufficient to show why $Pr(s_{init} \models \diamond\mathbf{tar}) < 1$ holds in order to indicate that the expected reward property is refuted. We can generate a counterexample, for instance, by finding a finite path that starts in s_{init} and ends in a state from which there is no \mathbf{tar} -state reachable. The rewards of \mathcal{M} do not have to be considered for this purpose. Hence, this trivial case is of no further interest in this work.

From now on, we assume that $Pr(s_{init} \models \diamond\mathbf{tar}) = 1$, i.e., a \mathbf{tar} -state will eventually be reached from s_{init} with probability 1. It can be shown that $Pr(s \models \diamond\mathbf{tar}) = 1$ holds for all states $s \in S$ that are reachable from s_{init} without visiting a \mathbf{tar} -state. Note that this applies for all relevant states, i.e., states that are visited on a path in $\Pi_{s_{init}}^{\diamond\mathbf{tar}}$. Furthermore, it is assumed that every state in the considered MRM is reachable from s_{init} . Whenever this is not the case, we can find the states that are not reachable from

s_{init} via standard reachability analysis on the graph $\mathcal{G}_{\mathbf{P}}$. As these states are not relevant for the expected reward, we can remove them without affecting the expected rewards of the remaining states. We can conclude that from all states that are relevant for the expected reward, a **tar**-state will eventually be reached with probability 1. This directly follows from the assumption $Pr(s_{init} \models \diamond \mathbf{tar}) = 1$. Therefore, a critical subsystem of \mathcal{M} for $\mathbb{E}_{\triangleleft \lambda}(\diamond \mathbf{tar})$ does not have to show the details of how a **tar**-state will eventually be reached. Instead of that, transitions to states that are not part of the subsystem will be redirected to a new **tar**-state and interpreted as a “shortcut” to **tar**.

4.2 Critical Subsystems for Expected Reward Properties

Similar to the probabilistic case presented in [2, 11] as well as Chapter 3, a critical subsystem for an expected reward property is a (preferably small) part of the original system that already refutes the property. We will, again, use selections according to Definition 3.3 on page 18 in order to specify the set of selected states that are contained in a subsystem. We can assume $Pr^{\mathcal{M}}(s_{init} \models \diamond \mathbf{tar}) = 1$ holds for the original system \mathcal{M} with initial state s_{init} . On the other hand, $Pr^{\mathcal{M}'}(s_{init} \models \diamond \mathbf{tar}) = 1$ has to hold for every possible subsystem \mathcal{M}' to avoid that $ExpRew^{\mathcal{M}'}(s_{init} \models \diamond \mathbf{tar}) = \infty$ and therefore $ExpRew^{\mathcal{M}'}(s_{init} \models \diamond \mathbf{tar}) > ExpRew^{\mathcal{M}}(s_{init} \models \diamond \mathbf{tar})$ holds.

Definition 4.1 (Critical Subsystem for an Expected Reward Property)

Let $\mathcal{M} = (\mathcal{D}, rew)$ be an MRM with the underlying DTMC $\mathcal{D} = (S, \mathbf{P}, s_{init}, AP, L)$ such that $Pr^{\mathcal{M}}(s_{init} \models \diamond \mathbf{tar}) = 1$ for an atomic proposition $\mathbf{tar} \in AP$. Moreover, let S' be a selection, i.e., $s_{init} \in S' \subseteq S$. The *subsystem* of \mathcal{M} induced by S' is given by the MRM $\mathcal{M}' = (\mathcal{D}', rew')$ with the underlying DTMC $\mathcal{D}' = (S' \uplus \{s_t\}, \mathbf{P}', s_{init}, AP, L')$ where $s_t \notin S$ is a new state and

$$\begin{aligned} \bullet \mathbf{P}'(s, s') &= \begin{cases} \mathbf{P}(s, s') & \text{for } s, s' \in S' \\ \sum_{s'' \in S \setminus S'} \mathbf{P}(s, s'') & \text{for } s \in S' \text{ and } s' = s_t \\ 1 & \text{for } s = s' = s_t \\ 0 & \text{for } s = s_t \text{ and } s' \in S', \end{cases} \\ \bullet L'(s) &= \begin{cases} L(s) & \text{for } s \in S' \\ \{\mathbf{tar}\} & \text{for } s = s_t, \end{cases} \\ \text{and} \\ \bullet rew'(s) &= \begin{cases} rew(s) & \text{for } s \in S' \\ 0 & \text{for } s = s_t. \end{cases} \end{aligned}$$

The subsystem \mathcal{M}' is *critical* for a property $\mathbb{E}_{\triangleleft \lambda}(\diamond \mathbf{tar})$ if $\mathcal{M}' \not\models \mathbb{E}_{\triangleleft \lambda}(\diamond \mathbf{tar})$. ■

Transitions that, in the original system, lead to a non-selected state get redirected to a new target state s_t . As described above, these transitions can be seen as “shortcuts”

to a target state and allow us to hide the details of how such a state will eventually be reached. Moreover, the definition ensures that $Pr^{\mathcal{M}'}(s_{init} \models \diamond \mathbf{tar}) = 1$ holds for every subsystem \mathcal{M}' of \mathcal{M} .

Example 4.2

Consider the MRM \mathcal{M} of the communication protocol from Example 2.6, depicted in Figure 2.2 on page 15. Moreover, consider the expected reward property $\mathbb{E}_{<7}(\diamond \mathbf{end})$ from Example 2.5 on page 11. The subsystem \mathcal{M}' of \mathcal{M} induced by the selection $S' = \{s_{init}, c_1, o_1, a_1, c_2, a_2\}$ is depicted in Figure 4.1. For \mathcal{M}' it holds that

$$ExpRew^{\mathcal{M}'}(s_{init} \models \diamond \mathbf{end}) = 7.364.$$

Hence, \mathcal{M}' is critical for the property $\mathbb{E}_{<7}(\diamond \mathbf{end})$. ■

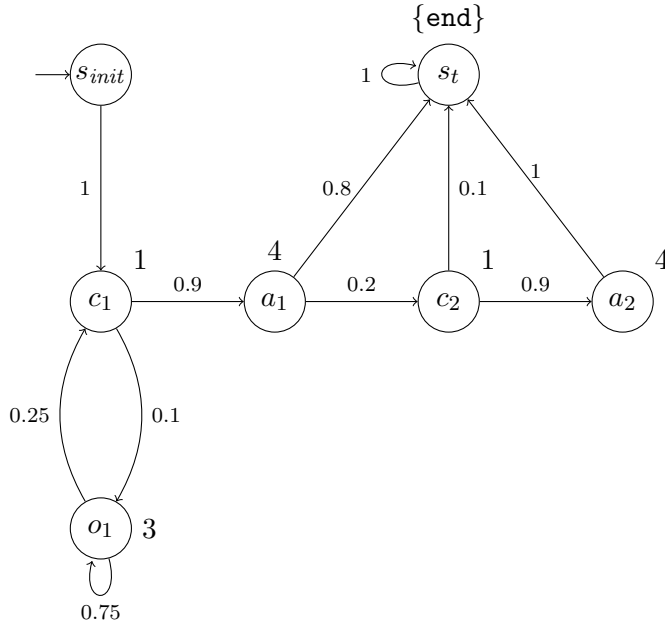


Figure 4.1: Critical subsystem of the MRM from Figure 2.2 for the property $\mathbb{E}_{<7}(\diamond \mathbf{end})$.

4.3 Alternative Definition of Critical Subsystems

According to Definition 4.1, a critical subsystem for expected reward properties is restricted based on the selected states, i.e., non-selected states are disregarded. We also want to present an alternative definition of critical subsystems where the restriction is based on the reward assignment. To avoid confusion, this is called a critical reward subsystem.

Definition 4.3 (Critical Reward Subsystem)

Let $\mathcal{M} = (\mathcal{D}, rew)$ be an MRM with the underlying DTMC $\mathcal{D} = (S, \mathbf{P}, s_{init}, AP, L)$ such that $Pr(s_{init} \models \diamond \mathbf{tar}) = 1$ for an atomic proposition $\mathbf{tar} \in AP$. For a set of selected states $S' \subseteq S$, the *reward subsystem* of \mathcal{M} induced by S' is defined as the MRM $\mathcal{M}' = (\mathcal{D}, rew')$ where

$$rew'(s) = \begin{cases} rew(s) & \text{for } s \in S' \\ 0 & \text{otherwise.} \end{cases}$$

The reward subsystem \mathcal{M}' is *critical* for a property $\mathbb{E}_{\leq \lambda}(\diamond \mathbf{tar})$ if $\mathcal{M}' \not\models \mathbb{E}_{\leq \lambda}(\diamond \mathbf{tar})$. ■

A critical reward subsystem indicates the parts of the system that give enough reward to refute a property, without ruling out possible system behavior (i.e., without disregarding states in the underlying DTMC). Depending on the particular application, this can be advantageous, although there are examples where a critical subsystem according to Definition 4.1 should be preferred. For instance, consider an MRM $\mathcal{M} = (\mathcal{D}, rew)$ with exactly one state s such that $rew(s) > 0$. Every reward subsystem of \mathcal{M} is either equal to \mathcal{M} or only has states with reward 0 and is therefore less useful for debugging purposes.

4.4 Reduction from Reachability Properties

In this section, we are going to show that the problem of finding a critical subsystem for a reachability property can be reduced to finding a critical subsystem for an expected reward property (according to Definition 4.1). This allows us to use methods designed for expected reward properties in order to generate critical subsystems for reachability properties as seen in Section 3.2. For this purpose, the given DTMC is transformed to an MRM where reward 1 is assigned to every target state and the outgoing transitions of such states are redirected to a new state s_{done} . Moreover, a new target label \mathbf{tar}' is introduced and assigned to s_{done} as well as to all states from which an old target state is not reachable. We call this transformation the reachability reward construction.

Definition 4.4 (Reachability Reward Construction)

Let $\mathcal{D} = (S, \mathbf{P}, s_{init}, AP, L)$ be a DTMC, $\mathbf{tar} \in AP$ and $T \subseteq S$ the set of \mathbf{tar} -states. The *reachability reward construction* of \mathcal{D} with respect to \mathbf{tar} is given by the MRM $\tilde{\mathcal{M}} = (\tilde{\mathcal{D}}, \tilde{rew})$ with the underlying DTMC $\tilde{\mathcal{D}} = (S \uplus \{s_{done}\}, \tilde{\mathbf{P}}, s_{init}, AP \uplus \{\mathbf{tar}'\}, \tilde{L})$ where

$$\begin{aligned} \bullet \tilde{\mathbf{P}}(s, s') &= \begin{cases} 1 & \text{for } s \in T \cup \{s_{done}\} \text{ and } s' = s_{done} \\ 0 & \text{for } s \in T \cup \{s_{done}\} \text{ and } s' \neq s_{done} \\ \mathbf{P}(s, s') & \text{otherwise,} \end{cases} \\ \bullet \tilde{L}(s) &= \begin{cases} \{\mathbf{tar}'\} & \text{for } s = s_{done} \\ L(s) \cup \{\mathbf{tar}'\} & \text{for } s \in S \text{ and no } \mathbf{tar}\text{-state is reachable from } s \text{ in } \mathcal{D} \\ L(s) & \text{otherwise,} \end{cases} \end{aligned}$$

and

$$\bullet \widetilde{rew}(s) = \begin{cases} 1 & \text{if } \mathbf{tar} \in L(s) \\ 0 & \text{otherwise.} \end{cases} \quad \blacksquare$$

Example 4.5 (Reachability Reward Construction and its Subsystem)

Consider the DTMC \mathcal{D} , depicted on the top left side of Figure 4.2. The reachability reward construction of \mathcal{D} with respect to \mathbf{tar} is given by the MRM $\widetilde{\mathcal{M}}$ which is depicted on the right top side of Figure 4.2. Moreover, the subsystems $\mathcal{D}_{S'}$ of \mathcal{D} and $\widetilde{\mathcal{M}}_{S'}$ of $\widetilde{\mathcal{M}}$ induced by the selection $S' = \{s_0, s_1\}$ are given by the models depicted at the bottom of Figure 4.2. We can see that

$$Pr^{\mathcal{D}_{S'}}(s_0 \models \diamond \mathbf{tar}) = 0.5 = ExpRew^{\widetilde{\mathcal{M}}_{S'}}(s_0 \models \diamond \mathbf{tar}'). \quad \blacksquare$$

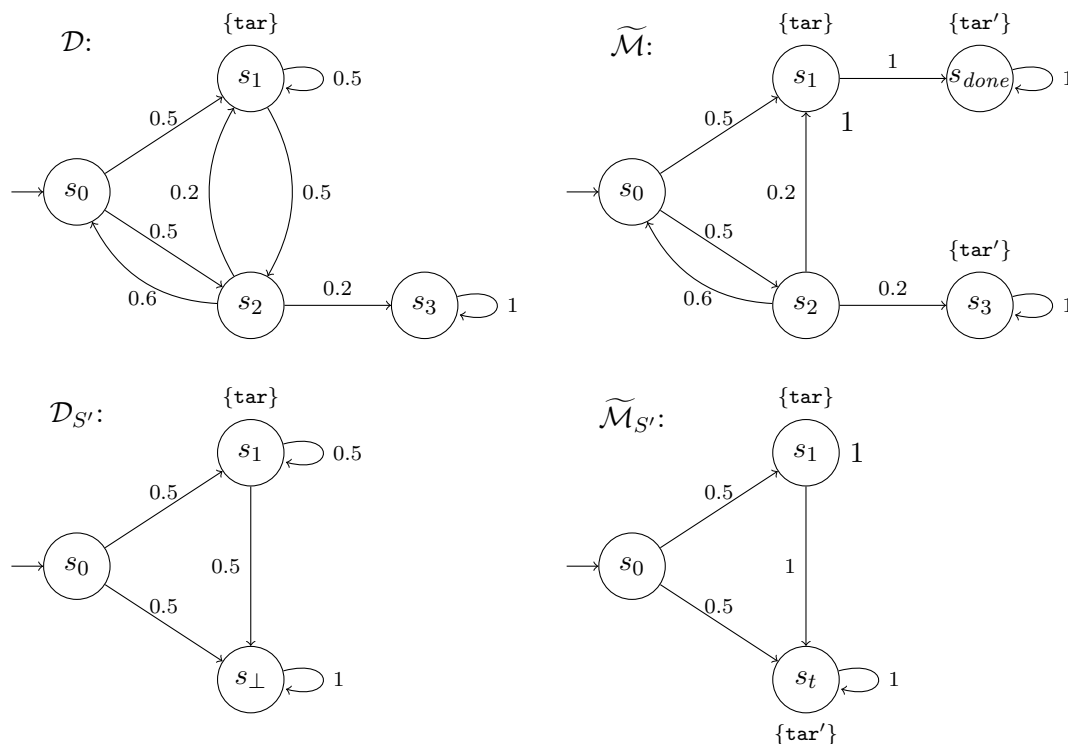


Figure 4.2: Reachability reward construction $\widetilde{\mathcal{M}}$ (top right) of the DTMC \mathcal{D} (top left) with respect to \mathbf{tar} . The Subsystems of both models induced by the selection $S' = \{s_0, s_1\}$ are depicted at the bottom.

Note that for every DTMC, the probability to reach a target state or a state from which no target state is reachable equates 1. For a reachability reward construction $\widetilde{\mathcal{M}}$ of a DTMC \mathcal{D} , all outgoing transitions of target states are redirected to a \mathbf{tar}' state. Hence,

$$Pr^{\widetilde{\mathcal{M}}}(s_{init} \models \diamond \mathbf{tar}') = 1.$$

We are now going to prove the following lemma which intuitively states that a critical subsystem of a reachability reward constructions is consistent with a critical subsystem of the original DTMC.

Lemma 4.6

Let $\widetilde{\mathcal{M}}$ be the reachability reward construction of a DTMC $\mathcal{D} = (S, \mathbf{P}, s_{init}, AP, L)$ with respect to $\mathbf{tar} \in AP$ and let $\mathbb{P}_{\triangleleft\lambda}(\diamond\mathbf{tar})$ be a reachability property. A selection $S' \subseteq S$ induces a critical subsystem of $\widetilde{\mathcal{M}}$ for $\mathbb{E}_{\triangleleft\lambda}(\diamond\mathbf{tar}')$ if and only if S' induces a critical subsystem of \mathcal{D} for $\mathbb{P}_{\triangleleft\lambda}(\diamond\mathbf{tar})$. ■

Proof: For the DTMC $\mathcal{D} = (S, \mathbf{P}, s_{init}, AP, L)$ and $\mathbf{tar} \in AP$, let $\widetilde{\mathcal{M}} = (\widetilde{\mathcal{D}}, \widetilde{rew})$ be the reachability reward construction. Furthermore consider the reachability property $\mathbb{P}_{\triangleleft\lambda}(\diamond\mathbf{tar})$ and an arbitrary selection S' with $s_{init} \in S' \subseteq S$. Let $\mathcal{D}_{S'}$ and $\widetilde{\mathcal{M}}_{S'} = (\widetilde{\mathcal{D}}_{S'}, \widetilde{rew}_{S'})$ be the subsystems of \mathcal{D} and $\widetilde{\mathcal{M}}$ induced by S' . The corresponding transition probability functions are denoted by $\mathbf{P}_{S'}$ and $\widetilde{\mathbf{P}}_{S'}$, respectively. We are going to show that

$$ExpRew^{\widetilde{\mathcal{M}}_{S'}}(s_{init} \models \diamond\mathbf{tar}') = Pr^{\mathcal{D}_{S'}}(s_{init} \models \diamond\mathbf{tar})$$

holds. Thus, $\widetilde{\mathcal{M}}_{S'}$ is critical for $\mathbb{E}_{\triangleleft\lambda}(\diamond\mathbf{tar}')$ if and only if $\mathcal{D}_{S'}$ is critical for $\mathbb{P}_{\triangleleft\lambda}(\diamond\mathbf{tar})$.

We will begin with a few auxiliary statements. Let $\Pi_{s_{init}}^{\diamond\mathbf{tar}'}$ refer to the set of paths in the the subsystem $\widetilde{\mathcal{M}}_{S'}$ that start in s_{init} and end at the first visit of a \mathbf{tar}' -state. Furthermore, let $\Pi_1 \subseteq \Pi_{s_{init}}^{\diamond\mathbf{tar}'}$ be the set of those paths that visit a \mathbf{tar} -state while $\Pi_2 \subseteq \Pi_{s_{init}}^{\diamond\mathbf{tar}'}$ is the set of paths that never visit a \mathbf{tar} -state. Formally,

$$\begin{aligned} \Pi_1 &:= \{s_0 \dots s_n \in \Pi_{s_{init}}^{\diamond\mathbf{tar}'} \mid \exists i \in \{0, \dots, n\}. \mathbf{tar} \in \widetilde{L}(s_i)\} \text{ and} \\ \Pi_2 &:= \Pi_{s_{init}}^{\diamond\mathbf{tar}'} \setminus \Pi_1. \end{aligned}$$

Since $\widetilde{rew}_{S'}(s) = 0$ for all states s with $\mathbf{tar} \notin \widetilde{L}(s)$, it follows that

$$\widetilde{rew}_{S'}(\hat{\pi}) = 0 \text{ for all } \hat{\pi} \in \Pi_2. \quad (1)$$

For every \mathbf{tar} -state in $\widetilde{\mathcal{M}}$ there is only one outgoing transition which leads to s_{done} . Moreover, $s_{done} \notin S$ will never be selected in S' . Hence, a path in $\widetilde{\mathcal{M}}_{S'}$ can visit a state s with $\mathbf{tar} \in \widetilde{L}(s)$ at most once and, with probability 1, the next state will be the absorbing \mathbf{tar}' -state s_t of the subsystem $\widetilde{\mathcal{M}}_{S'}$. For every path $s_0 \dots s_n \in \Pi_1$ we can conclude

$$\mathbf{tar} \in \widetilde{L}(s_{n-1}), s_n = s_t \text{ and } \mathbf{tar} \notin \widetilde{L}(s_i) \text{ for all } i \neq n-1, \quad (2)$$

$$\widetilde{rew}_{S'}(s_0 \dots s_n) = \widetilde{rew}_{S'}(s_{n-1}) = 1, \text{ and} \quad (3)$$

$$\widetilde{\mathbf{P}}_{S'}(s_{n-1}, s_n) = 1. \quad (4)$$

Furthermore, a \mathbf{tar} -state in $\widetilde{\mathcal{M}}_{S'}$ is only reachable via states $s \in S'$ with $\mathbf{tar} \notin L(s)$. Therefore, the used transitions can be taken in $\mathcal{D}_{S'}$ with the same probability, meaning

that

$$\tilde{\mathbf{P}}_{S'}(s_i, s_{i+1}) = \mathbf{P}(s_i, s_{i+1}) = \mathbf{P}_{S'}(s_i, s_{i+1}) \quad (5)$$

for every path $s_0 \dots s_n \in \Pi_1$ and for all $0 \leq i < n - 1$. At last, we want to show that

$$s_0 \dots s_n \in \Pi_1 \Leftrightarrow s_0 \dots s_{n-1} \in \Pi_{s_{init}}^{\diamond \mathbf{tar}} \text{ and } s_n = s_t, \quad (6)$$

where $\Pi_{s_{init}}^{\diamond \mathbf{tar}}$ refers to paths in $\mathcal{D}_{S'}$. “ \Rightarrow ” directly follows from (2) and (5). “ \Leftarrow ” holds since a path $s_0 \dots s_{n-1} \in \Pi_{s_{init}}^{\diamond \mathbf{tar}}$ only visits states in S' and is therefore also a valid path in $\tilde{\mathcal{M}}_{S'}$. Note that in the subsystem $\tilde{\mathcal{M}}_{S'}$, there is only one successor of the \mathbf{tar} -state s_{n-1} , namely s_t . Thus $s_0 \dots s_{n-1} s_t \in \Pi_1$.

Since $Pr^{\tilde{\mathcal{M}}}(s_{init} \models \diamond \mathbf{tar}') = 1$, it also holds that $Pr^{\tilde{\mathcal{M}}_{S'}}(s_{init} \models \diamond \mathbf{tar}') = 1$ and we can exclude the case $ExpRew^{\tilde{\mathcal{M}}_{S'}}(s_{init} \models \diamond \mathbf{tar}') = \infty$. With (1)-(6) and $\Pi_{s_{init}}^{\diamond \mathbf{tar}'} = \Pi_1 \uplus \Pi_2$, we can proof the following:

$$\begin{aligned} & ExpRew^{\tilde{\mathcal{M}}_{S'}}(s_{init} \models \diamond \mathbf{tar}') \\ \stackrel{\text{def.}}{=} & \sum_{\hat{\pi} \in \Pi_{s_{init}}^{\diamond \mathbf{tar}'}} \tilde{\mathbf{P}}_{S'}(\hat{\pi}) \cdot \widetilde{rew}_{S'}(\hat{\pi}) \\ = & \sum_{\hat{\pi} \in \Pi_1} \tilde{\mathbf{P}}_{S'}(\hat{\pi}) \cdot \underbrace{\widetilde{rew}_{S'}(\hat{\pi})}_{\stackrel{(3)}{=} 1} + \sum_{\hat{\pi} \in \Pi_2} \tilde{\mathbf{P}}_{S'}(\hat{\pi}) \cdot \underbrace{\widetilde{rew}_{S'}(\hat{\pi})}_{\stackrel{(1)}{=} 0} \\ = & \sum_{s_0 \dots s_n \in \Pi_1} \tilde{\mathbf{P}}_{S'}(s_0 \dots s_{n-1}) \cdot \underbrace{\tilde{\mathbf{P}}_{S'}(s_{n-1}, s_n)}_{\stackrel{(4)}{=} 1} \\ \stackrel{(5)}{=} & \sum_{s_0 \dots s_n \in \Pi_1} \mathbf{P}_{S'}(s_0 \dots s_{n-1}) \\ \stackrel{(6)}{=} & \sum_{s_0 \dots s_{n-1} \in \Pi_{s_{init}}^{\diamond \mathbf{tar}}} \mathbf{P}_{S'}(s_0 \dots s_{n-1}) \\ \stackrel{\text{def.}}{=} & Pr^{\mathcal{D}_{S'}}(s_{init} \models \diamond \mathbf{tar}) \quad \square \end{aligned}$$

In the next chapter, we will see how critical subsystems for expected rewards can be generated. These approaches also yield methods to generate critical subsystems for reachability properties by applying Lemma 4.6.

Chapter 5

Generating Critical Subsystems

We now introduce two heuristic approaches to generate critical subsystems for expected reward properties. The focus lies on finding small critical subsystems within an acceptable amount of time. Although these algorithms are designed for subsystems according to Definition 4.1 on page 26, they can easily be modified to generate critical reward subsystems according to Definition 4.3 on page 28.

5.1 Path Search Approach

The first idea is to extend the local path search presented in [11] and Section 3.2. Besides the probability of a path, the cumulated reward should also be considered. Basically, this is done in terms of a value function that takes probabilities as well as rewards into account. States that are already selected or labeled with the target label are connected via most valuable paths (instead of most probable paths).

5.1.1 The Basic Algorithm

Let $\mathcal{M} = (\mathcal{D}, rew)$ be a finite MRM with $\mathcal{D} = (S, \mathbf{P}, s_{init}, AP, L)$ such that every **tar**-state of \mathcal{M} is absorbing and $Pr(s_{init} \models \diamond \mathbf{tar}) = 1$. Assume that we want to generate a critical subsystem of \mathcal{M} for the expected reward property $\mathbb{E}_{\diamond \lambda}(\diamond \mathbf{tar})$. Recall that $\Delta_{S_{start}}^{S_{end}}$ is the set of paths that connect the two sets of states $S_{start} \subseteq S$ and $S_{end} \subseteq S$, excluding intermediate visits of $S_{start} \cup S_{end}$ and paths that never leave S_{start} . Note that the sets S_{start} and S_{end} do not have to be disjoint. Hence, a path in $\Delta_{S_{start}}^{S_{end}}$ may start in a state $s \in S_{end}$ providing that s is also contained in the set S_{start} . We are going to use a value function¹ $\mathbf{V}: S \times S \rightarrow [0, 1]$ and will refer to paths in the graph $\mathcal{G}_{\mathbf{V}} = (S, E)$ with $(s, s') \in E$ if and only if $\mathbf{V}(s, s') > 0$. Let us first consider the problem of finding

¹Different approaches for a reasonable definition of the value function will be presented later.

a most valuable path that connects $S_{start} \subseteq S$ with $S_{end} \subseteq S$. This is a path $\hat{\pi} \in \Delta_{S_{start}}^{S_{end}}$ such that

$$\mathbf{V}(\hat{\pi}) \geq \mathbf{V}(\hat{\sigma}) \text{ for all } \hat{\sigma} \in \Delta_{S_{start}}^{S_{end}},$$

where $\mathbf{V}(\hat{\pi})$ is the value of the path $\hat{\pi}$, i.e.,

$$\mathbf{V}(s_0 \dots s_n) := \prod_{i=0}^{n-1} \mathbf{V}(s_i, s_{i+1}).$$

This is analogous to a most probable path that connects two sets of states in a DTMC, with the exception that the value function \mathbf{V} is considered instead of the transition probability function \mathbf{P} .

If there is a most valuable path connecting S_{start} with S_{end} , Algorithm 1 depicts how such a path can be found. The procedure is similar to the well known shortest path algorithm by Dijkstra [6]. However, instead of searching a path such that the sum of the weights is minimal, a path is searched such that the product of the values is maximal. Moreover, the initialization has been modified in order to exclude paths that are not in $\Delta_{S_{start}}^{S_{end}}$.

At first, the values of a most valuable path from S_{start} to $\{s'\}$ are computed for all states $s' \in S$ and stored in the array *maxValue* (Line 2 to 27). This is done by initializing the entries *maxValue*[s'] for all $s' \in S \setminus S_{start}$ with the value of a most valuable path of length 1 from S_{start} to $\{s'\}$, more precisely,

$$\mathit{maxValue}[s'] = \begin{cases} \max_{s \in S_{start}} \mathbf{V}(s, s') & \text{if } s' \in S \setminus S_{start} \\ 0 & \text{otherwise} \end{cases}$$

(Line 2 to 14). Furthermore, the set $S_{explore}$ contains all states that have to be explored. Initially, this equates to the set $\{s' \in S \mid \mathit{maxValue}[s'] > 0\}$ (Line 11). As long as there are states to explore, a state $s \in S_{explore}$ where $\mathit{maxValue}[s]$ is maximal is chosen, removed from $S_{explore}$, and explored (Line 15 to 27). Exploring a state s means that for all states $s' \in S \setminus \{s\}$ it is checked whether the currently known entry *maxValue*[s'] can be improved by considering the most valuable path from S_{start} to $\{s\}$ that has been found so far and extending it with s' . If this is the case, the entry *maxValue*[s'] is updated and s' is added to $S_{explore}$ (unless s' is contained in S_{start}). Besides the value of a most valuable path to a state s' , the preceding state of s' on that path is stored as well (Line 10 and 21). At Line 28, a state $s \in S_{end}$ is chosen, such that the value of a most valuable path from S_{start} to $\{s\}$ is maximal. After that, the resulting path $\hat{\pi}$ is constructed and returned. (Line 29 to 34).

Algorithm 2 depicts the generation of a critical subsystem of \mathcal{M} for $\mathbb{E}_{\diamond\lambda}(\diamond\mathbf{tar})$. The first step is to initialize all information regarding the value function (Line 1). Details to this are discussed later. After that, a most valuable path from the initial state to one of the **tar**-states is searched and the selection S' is initialized with the states visited on that path (Line 2 to 4). As long as the subsystem of \mathcal{M} induced by the selection

Algorithm 1 Most Valuable Path Search**Input:** Value function \mathbf{V} , set of all states S , subsets $S_{start} \subseteq S$ and $S_{end} \subseteq S$ **Output:** Most valuable path connecting S_{start} with S_{end} (if such a path exists)

```

1: procedure FINDMOSTVALUABLEPATH( $\mathbf{V}, S, S_{start}, S_{end}$ )
2:   for all  $s \in S$  do
3:      $maxValue[s] \leftarrow 0$ 
4:   end for
5:    $S_{explore} \leftarrow \emptyset$ 
6:   for all  $s \in S_{start}$  do
7:     for all  $s' \in S \setminus S_{start}$  do
8:       if  $maxValue[s'] < \mathbf{V}(s, s')$  then
9:          $maxValue[s'] \leftarrow \mathbf{V}(s, s')$ 
10:         $predecessor[s'] \leftarrow s$ 
11:         $S_{explore} \leftarrow S_{explore} \cup \{s'\}$ 
12:      end if
13:    end for
14:  end for
15:  while  $S_{explore} \neq \emptyset$  do
16:    chose  $s \in S_{explore}$  with  $maxValue[s] \geq maxValue[s']$  for all  $s' \in S_{explore}$ 
17:     $S_{explore} \leftarrow S_{explore} \setminus \{s\}$ 
18:    for all  $s' \in S \setminus \{s\}$  do
19:      if  $maxValue[s'] < maxValue[s] \cdot \mathbf{V}(s, s')$  then
20:         $maxValue[s'] \leftarrow maxValue[s] \cdot \mathbf{V}(s, s')$ 
21:         $predecessor[s'] \leftarrow s$ 
22:        if  $s' \notin S_{start}$  then
23:           $S_{explore} \leftarrow S_{explore} \cup \{s'\}$ 
24:        end if
25:      end if
26:    end for
27:  end while
28:  chose  $s \in S_{end}$  with  $maxValue[s] \geq maxValue[s']$  for all  $s' \in S_{end}$ 
29:   $\hat{\pi} \leftarrow s$ 
30:  repeat
31:     $s \leftarrow predecessor[s]$ 
32:     $\hat{\pi} \leftarrow s\hat{\pi}$ 
33:  until  $s \notin S_{start}$ 
34:  return  $\hat{\pi}$ 
35: end procedure

```

S' (depicted by $\text{SUBSYS}(\mathcal{M}, S')$) is not critical for the given property, S' is extended by the states visited on a most valuable path that connects either two already selected states or a selected state with a **tar**-state (Line 5 to 8). Since these paths always visit non-selected states, more states get selected in every iteration until $\text{SUBSYS}(\mathcal{M}, S')$ eventually becomes critical.

Algorithm 2 Path Search Critical Subsystem Generation

Input: Finite MRM $\mathcal{M} = (\mathcal{D}, \text{rew})$ with $\mathcal{D} = (S, \mathbf{P}, s_{\text{init}}, AP, L)$, absorbing **tar**-states, $\mathbb{E}_{\triangleleft\lambda}(\diamond\text{tar})$ such that $\mathcal{M} \not\models \mathbb{E}_{\triangleleft\lambda}(\diamond\text{tar})$, $Pr(s_{\text{init}} \models \diamond\text{tar}) = 1$ and $\text{tar} \notin L(s_{\text{init}})$
Output: A critical subsystem of \mathcal{M} for $\mathbb{E}_{\triangleleft\lambda}(\diamond\text{tar})$

```

1: initialize value function  $\mathbf{V}: S \times S \rightarrow [0, 1]$ 
2:  $T \leftarrow \{s \in S \mid \text{tar} \in L(s)\}$ 
3:  $\hat{\pi} \leftarrow \text{FINDMOSTVALUABLEPATH}(\mathbf{V}, S, \{s_{\text{init}}\}, T)$ 
4:  $S' \leftarrow \{s \in S \mid s \text{ is visited by } \hat{\pi}\}$ 
5: while  $\text{SUBSYS}(\mathcal{M}, S') \not\models \mathbb{E}_{\triangleleft\lambda}(\diamond\text{tar})$  do
6:    $\hat{\pi} \leftarrow \text{FINDMOSTVALUABLEPATH}(\mathbf{V}, S, S', S' \cup T)$ 
7:    $S' \leftarrow S' \cup \{s \in S \mid s \text{ is visited by } \hat{\pi}\}$ 
8: end while
9: return  $\text{SUBSYS}(\mathcal{M}, S')$ 

```

If we equate the value function \mathbf{V} with the transition probability function \mathbf{P} , the presented algorithm corresponds to the local path search approach which has been presented in [11]. More information can be found there.

Example 5.1 (Path Search Approach for Communication Protocol)

Let \mathcal{M} be the MRM of the communication protocol from Example 2.6, depicted in Figure 2.2 on page 15. Consider the expected reward property $\mathbb{E}_{<7}(\diamond\text{end})$ from Example 2.5 on page 11 which is refuted in the initial state s_{init} of \mathcal{M} . We are going to apply the path search approach to generate a critical subsystem. Assume that the value function \mathbf{V} is given by

$$\mathbf{V}(s, s') = \frac{\text{ExpRew}(s \models \diamond\text{tar}) + \varepsilon}{\max_{s'' \in S} (\text{ExpRew}(s'' \models \diamond\text{tar})) + 2\varepsilon}$$

with $\varepsilon = 0.2$ and providing that $\mathbf{P}(s, s') > 0$. For $\mathbf{P}(s, s') = 0$, we set $\mathbf{V}(s, s') = 0$. We will discuss this and other value functions later. The graph $\mathcal{G}_{\mathbf{V}}$ is depicted in Figure 5.1 (top). Analogous to the depiction of DTMCs, we use labeled transitions between two states s and s' to depict the value $\mathbf{V}(s, s') > 0$.

At first, a most valuable path connecting $\{s_{\text{init}}\}$ with $\{\text{del}, \text{end}\}$ is searched. This path is unique and given by

$$\pi_0 := s_{\text{init}} c_1 a_1 \text{del}.$$

The selection $S' \subseteq S$ is initialized with the states visited on π_0 , i.e., $S' = \{s_{\text{init}}, c_1, a_1, \text{del}\}$. The subsystem \mathcal{M}_0 of \mathcal{M} induced by S' is depicted in Figure 5.1 (center left). Since

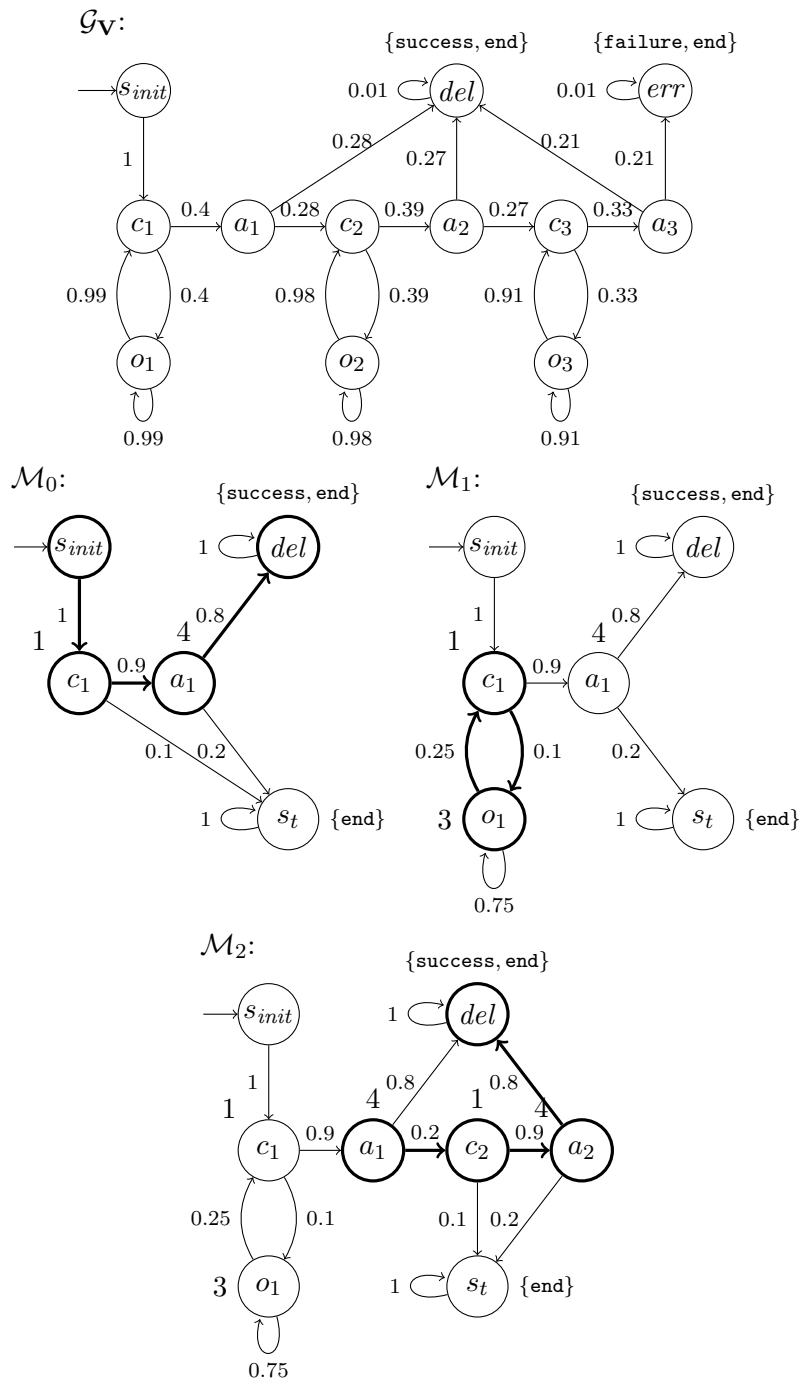


Figure 5.1: Graph \mathcal{G}_V (top) and the subsystems \mathcal{M}_0 , \mathcal{M}_1 and \mathcal{M}_2 (bottom) which illustrate the path search approach. The transitions and states along the recently found path are marked with thick lines.

$ExpRew^{\mathcal{M}_0}(s_{init} \models \diamond \text{end}) = 4.6 < 7$, the subsystem is not critical yet. Now we search a most valuable path connecting $\{s_{init}, c_1, a_1, del\}$ with $\{s_{init}, c_1, a_1, del, tar, \}$ which results in the path

$$\pi_1 := c_1 o_1 c_1.$$

The selection S' is extended to $S' = \{s_{init}, c_1, a_1, del, o_1\}$. For the resulting subsystem \mathcal{M}_1 (center right of Figure 5.1), it holds that $ExpRew^{\mathcal{M}_1}(s_{init} \models \diamond \text{end}) = 6.444 < 7$. Hence, another path is required. After finding the next path

$$\pi_2 := a_1 c_2 a_2 del$$

and adding the visited states to our selection, we have $S' = \{s_{init}, c_1, a_1, del, o_1, c_2, a_2\}$. The subsystem \mathcal{M}_2 (bottom of Figure 5.1) of \mathcal{M} induced by S' is critical for $\mathbb{E}_{<7}(\diamond \text{end})$, since $ExpRew^{\mathcal{M}_2}(s_{init} \models \diamond \text{end}) = 7.364 \geq 7$. ■

Requirements for the value function. Algorithm 1 does not work properly on arbitrary functions $\mathbf{V}: S \times S \rightarrow \mathbb{R}$. This is illustrated in the following example.

Example 5.2 (Malfunctioning Value Functions)

Consider the graph $\mathcal{G}_{\mathbf{V}}$ depicted in Figure 5.2. All paths in $\Delta_{\{s_0\}}^{\{s_2\}}$ are of the form $s_0 (s_1)^k s_2$ and have the value v^k for $k \in \mathbb{N}$. If $v > 1$, this value increases with the length of the path. Thus, there is no most valuable path that can be found by the algorithm. For $v = 1$, every path has the value $1^k = 1$ and therefore there are infinitely many most valuable paths. Although Algorithm 1 rules out such 1-valued loops by using the strict operator $<$ in Line 8 and 19, this case can still be problematic due to rounding errors if the used arithmetic is not exact. Negative values should be excluded as well. For instance, consider $v = -0.5$. To find a most valuable path connecting $\{s_0\}$ with $\{s_3\}$, Algorithm 1 finds the (unique) most valuable path $s_0 s_1 s_2$ from $\{s_0\}$ to $\{s_2\}$ and extends it by the state s_3 . This yields a path with value -1 , although the path $s_0 s_1 s_1 s_2 s_3 \in \Delta_{\{s_0\}}^{\{s_3\}}$ would have a higher value of 0.5 . ■

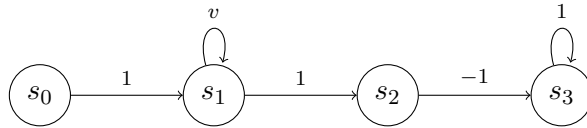


Figure 5.2: Exemplary graph $\mathcal{G}_{\mathbf{V}}$ to illustrate when a malfunctioning value function.

We conclude that for all $s, s' \in S$, the value function \mathbf{V} has to meet the conditions

- $0 \leq \mathbf{V}(s, s') \leq 1$,
- $\mathbf{V}(s, s') = 1$ implies $\mathbf{V}(s, s'') = 0$ for all $s'' \neq s'$ and
- $\mathbf{V}(s, s') > 0$ if and only if $\mathbf{P}(s, s') > 0$.

The last condition ensures that every path in the graph $\mathcal{G}_{\mathbf{V}}$ is also a valid path in \mathcal{M} and vice versa. Hence, every path found by Algorithm 1 only visits a **tar**-state if it is the last state on that path (recall that all **tar**-states of \mathcal{M} are absorbing). Moreover, if a state is relevant for the considered property (i.e., it can contribute to a critical subsystem), it will eventually be selected at Line 7 of Algorithm 2 (unless a critical subsystem has been found before).

5.1.2 Modifications of the Algorithm

We will now discuss different modifications of Algorithm 1 and 2. Besides different value functions, this includes altering the set of target states and weighting the value of a paths. Reasonably combined, these modifications can be helpful for finding small critical subsystems within acceptable time. Let $\mathcal{M} = (\mathcal{D}, rew)$ with $\mathcal{D} = (S, \mathbf{P}, s_{init}, AP, L)$ as well as $\mathbb{E}_{\triangleleft\lambda}(\diamond\mathbf{tar})$ be the input of Algorithm 2. Moreover, let \mathbf{V} , S , S_{start} and S_{end} be the input of Algorithm 1.

The value function. In this work, we discuss three different value functions, namely \mathbf{V}_1 , \mathbf{V}_2 , and \mathbf{V}_3 . For $i \in \{1, 2, 3\}$ we set

$$\mathbf{V}_i : S \times S \rightarrow [0, 1].$$

The first value function is given by

$$\mathbf{V}_1(s, s') = \mathbf{P}(s, s') \cdot \frac{rew(s) + \varepsilon}{\max_{s'' \in S}(rew(s'')) + \varepsilon}$$

for a sufficiently small $\varepsilon > 0$. Basically, the probability of moving from state s to s' is multiplied with the reward assigned to the state s . However, the rewards are divided by the maximal reward that occurs in the MRM, which ensures $\mathbf{V}_1(s, s') \leq 1$. Moreover ε is added to the numerator to avoid that $\mathbf{V}_1(s, s') = 0$ whenever $rew(s) = 0$. We also add ε to the denominator which avoids division by 0 or $\mathbf{V}_1(s, s') > 1$ if $rew(s) = \max_{s'' \in S}(rew(s''))$. This function provides a balance between probability of a transition and reward of a state. According to \mathbf{V}_1 , paths that use transitions with high probability and visit states with high rewards will have a high value.

Another value function uses the expected reward of every state. This data can be computed by solving a linear equation system (see Chapter 2) and is usually obtained as a side-product from model checking $\mathcal{M} \models \mathbb{E}_{\triangleleft\lambda}(\diamond\mathbf{tar})$. We define the second value function as

$$\mathbf{V}_2(s, s') = \frac{ExpRew(s \models \diamond\mathbf{tar}) + \varepsilon}{\max_{s'' \in S}(ExpRew(s'' \models \diamond\mathbf{tar})) + 2\varepsilon}$$

for a sufficiently small $\varepsilon > 0$ and providing that $\mathbf{P}(s, s') > 0$. For $\mathbf{P}(s, s') = 0$ we set $\mathbf{V}(s, s') = 0$. In order to fulfill the requirements for a value function we need to divide with the maximal occurring value and add ε in the numerator. In the denominator, we

add 2ε which ensures that $\mathbf{V}_2(s, s') < 1$ and therefore the requirement “ $\mathbf{V}(s, s') = 1$ implies $\mathbf{V}(s, s'') = 0$ for all $s'' \neq s'$ ” holds. The expected reward of a state provides information about the reward of the state itself as well as rewards of successive states, weighted with the probability to reach them. Hence, the value of a path according to \mathbf{V}_2 also evaluates the benefit of the successors of the visited states. This can be advantageous compared to the function \mathbf{V}_1 , which only considers the probabilities and rewards along a single path.

The last function we want to consider is given by

$$\mathbf{V}_3(s, s') = \mathbf{P}(s, s').$$

Providing that there is no other modification, the rewards will be ignored and the given algorithm corresponds to the local path searched presented in Chapter 3.

Using states with positive rewards as target states. As discussed in Chapter 4, the reachability of a **tar**-state does not have to be indicated by the subsystem. On the other hand, the reachability of states with (preferably high) rewards is of more interest. It might be useful to search for most valuable paths that connect already selected states with states that are equipped with positive rewards. This can be done by changing Line 2 of Algorithm 2 to:

$$T \leftarrow \{s \in S \mid \text{rew}(s) > 0 \text{ and } \mathbf{tar} \notin L(s)\}.$$

Since **tar**-states are absorbing, they do not have to be handled explicitly. It should be noted that a path found by Algorithm 1 may start in a state $s \in S_{end}$, providing that $s \in S_{start}$. Hence, this modification does not exclude paths that start in a state in T , i.e., paths that start in a state with positive reward.

Weighting the value of a path. Furthermore, Algorithm 1 can be adapted such that the value of a path is weighted with functions $v, w: S \rightarrow \mathbb{R}$, which depend on the first state and on the last state of the path. Then, a path $s_0 \dots s_n \in \Delta_{S_{start}}^{S_{end}}$ is searched, such that

$$v(s_0) \cdot \mathbf{V}(s_0 \dots s_n) \cdot w(s_n) \geq v(s'_0) \cdot \mathbf{V}(s'_0 \dots s'_n) \cdot w(s'_n) \text{ for all } s'_0 \dots s'_n \in \Delta_{S_{start}}^{S_{end}}.$$

This can be achieved by changing the Line 8, 9 and 28 of Algorithm 1 to

if $\text{maxValue}[s'] < v(s) \cdot \mathbf{V}(s, s')$ **then**

$\text{maxValue}[s'] \leftarrow v(s) \cdot \mathbf{V}(s, s')$

⋮

chose $s \in S_{end}$ with $\text{maxValue}[s] \cdot w(s) \geq \text{maxValue}[s'] \cdot w(s')$ for all $s' \in S_{end}$.

For $s \neq s_{init}$, $v(s)$ can be defined as the value of a most valuable path connecting $\{s_{init}\}$ with $\{s\}$. For $s = s_{init}$ we set $v(s) = 1$. The required information can be obtained from the first call of `FINDMOSTVALUABLEPATH` in Line 3 of Algorithm 2. With this modification, the value of a path $s_0 \dots s_n$ is decreased if every path from s_{init} to s_0 has a low value. This can reduce the number of selected states that are only reachable via low valued paths and therefore less beneficial for the critical subsystem.

If the states with positive rewards are used as target states, a reasonable definition of $w(s)$ is given by

$$w(s) = 1 + \frac{rew(s)}{\max_{s' \in S}(rew(s'))}.$$

This way, paths to states with high rewards are superior to paths that connect already selected states.

5.1.3 Performance Observations and Improvements

Time consumption. To check whether the subsystem induced by the current selection is critical (see Algorithm 2, Line 5), the subsystem has to be build and the given property has to be checked on it. Doing so in every iteration is very time intensive. Although the size of the returned subsystem may increase, it is reasonable to occasionally omit this step in order to save time. For instance, the condition can only be checked if at least $|S| \cdot c$ additional states have been selected since the last check. Here, $|S|$ denotes the number of states in the original system and c is a constant with $0 < c \leq 1$ that controls the size of the result and the run-time of the algorithm in the following way:

- The resulting critical subsystem can be up to $(|S| \cdot c) - 1$ states larger than necessary.
- The subsystem is build and model checking is performed at most $1/c$ times.

This method is used for the experiments we are going to present in Chapter 6.

Memory usage. In the discussed approach, the main factors for memory usage are given by the values of \mathbf{V} (if they are computed in advance), the subsets $T, S', S_{explore} \subseteq S$, the subsystem of \mathcal{M} induced by S' and the arrays *maxValue* and *predecessor*. Hence, the required memory is linear in the size of \mathcal{M} . In fact, practical tests have shown that parsing of \mathcal{M} or checking whether $\mathcal{M} \models \mathbb{E}_{\langle \lambda \rangle}(\diamond \mathbf{tar})$ is similarly, if not more, memory intensive. We will therefore exclude memory usage from the practical analysis presented in Chapter 6.

5.2 Best-first Search Approach

Another approach to generate critical subsystems is related to the extended best first search presented in [2] and Section 3.2. Again, a function $f: S \rightarrow \mathbb{R}$ evaluates how

beneficial it is to select a given state. However, this value function uses more information which is not necessarily given in the already explored system. On the one hand, this increases the effort to get the values of the function but, on the other hand, the provided values are more accurate which results in small critical subsystems. For the best-first search, we denote value functions with f (instead of \mathbf{V}) to avoid confusion with the value functions of the path search approach.

5.2.1 The Algorithm

The algorithm for the best-first search approach is depicted in Algorithm 3.

Algorithm 3 Best-first Search Critical Subsystem Generation

Input: Finite MRM $\mathcal{M} = (\mathcal{D}, rew)$ with $\mathcal{D} = (S, \mathbf{P}, s_{init}, AP, L)$,
 $\mathbb{E}_{\triangleleft\lambda}(\diamond\mathbf{tar})$ such that $\mathcal{M} \not\models \mathbb{E}_{\triangleleft\lambda}(\diamond\mathbf{tar})$ and $Pr(s_{init} \models \diamond\mathbf{tar}) = 1$
Output: A critical subsystem of \mathcal{M} for $\mathbb{E}_{\triangleleft\lambda}(\diamond\mathbf{tar})$

```

1: initialize function  $f: S \rightarrow \mathbb{R}$ 
2:  $S' \leftarrow \emptyset$ 
3:  $S_{explore} \leftarrow \{s_{init}\}$ 
4: repeat
5:   chose  $s \in S_{explore}$  with  $f(s) \geq f(s')$  for all  $s' \in S_{explore}$ 
6:    $S' \leftarrow S' \cup \{s\} \cup \{s' \in S \mid \mathbf{P}(s, s') > 0 \text{ and } \mathbf{tar} \in L(s')\}$ 
7:    $S_{explore} \leftarrow S_{explore} \setminus \{s\}$ 
8:    $S_{explore} \leftarrow S_{explore} \cup \{s' \in S \mid \mathbf{P}(s, s') > 0 \text{ and } s' \notin S'\}$ 
9: until SUBSYS( $\mathcal{M}, S'$ )  $\models \mathbb{E}_{\triangleleft\lambda}(\diamond\mathbf{tar})$ 
10: return SUBSYS( $\mathcal{M}, S'$ )

```

At first, the value function f is initialized (Line 1). Reasonable definitions of this function are proposed later. The set $S_{explore}$ always contains the states that are considered to be explored, starting with the initial state s_{init} (Line 3). In every iteration, a state $s \in S_{explore}$ with maximal value $f(s)$ is explored (Line 5 to 8). This means that s is added to the selection S' , removed from $S_{explore}$ and all non-selected successors of s are added to $S_{explore}$. \mathbf{tar} -states do not need to be explored. Whenever such a state is found, it is directly added to S' (Line 6) and therefore not added to $S_{explore}$. The generation stops as soon as the subsystem of \mathcal{M} induced by S' (denoted by SUBSYS(\mathcal{M}, S')) is critical for the given property (Line 9).

Similar to the previously introduced path search approach, it is not practical to build the subsystem and check the property in every iteration (see Section 5.1.3). Again, it is reasonable to only check this condition if at least $|S| \cdot c$ states have been added since the last check for a constant c with $0 < c \leq 1$.

Analogously, we can exclude memory usage from the practical analysis: Only the values of f (if they are computed in advance), the subsets $S', S_{explore} \subseteq S$ and the subsystem

of \mathcal{M} induced by S' have to be stored which is linear in the size of \mathcal{M} .

Example 5.3 (Best-first Search Approach for Communication Protocol)

Let \mathcal{M} be the MRM of the communication protocol from Example 2.6, depicted in Figure 2.2 on page 15. Again, we consider the expected reward property $\mathbb{E}_{<7}(\diamond\text{end})$ from Example 2.5 on page 11 which is refuted in the initial state s_{init} of \mathcal{M} . Figure 5.3 on page 44 illustrates the generation of a critical subsystem of \mathcal{M} for this property by using the value function $f(s) = \text{ExpRew}(s \models \diamond\text{end})$.

After every iteration, the states in the set S' as well as the transitions between them are depicted with solid lines while the states in S_{explore} and the transitions that lead to these states are depicted with dashed lines. Transition probabilities, atomic propositions as well as state rewards have been omitted. Instead of that, the value $f_1(s)$ is depicted next to every state $s \in S_{\text{explore}}$.

The initial state s_{init} is always the first state to be added to S' . In the next iteration, the state c_1 , which is the only successor of s_{init} , is chosen. While exploring c_1 , the states o_1 and a_1 have been found. Since $f_1(o_1) = 19.991 > 5.547 = f_1(a_1)$, the next state to be selected is o_1 . The procedure continues. After the 7th iteration, the set of selected states is given by $S' = \{s_{init}, c_1, o_1, a_1, c_2, o_2a_2\}$. For the subsystem \mathcal{M}' of \mathcal{M} induced by this selection it holds

$$\text{ExpRew}^{\mathcal{M}'}(s_{init} \models \diamond\text{end}) = 7.733 \not\leq 7.$$

Hence, the subsystem \mathcal{M}' is critical for $\mathbb{E}_{<7}(\diamond\text{end})$. ■

5.2.2 Different Value Functions

Let $\mathcal{M} = (\mathcal{D}, \text{rew})$ with $\mathcal{D} = (S, \mathbf{P}, s_{init}, AP, L)$ as well as $\mathbb{E}_{<\lambda}(\diamond\text{tar})$ be the input of Algorithm 3. Similar to the path search approach, we will discuss different value functions $f_i: S \rightarrow \mathbb{R}$ for $i \in \{1, 2, 3\}$. If a state s is beneficial to be part of a critical subsystem, the value $f_i(s)$ should be high.

The first value function we are going to consider is given by

$$f_1(s) = \text{ExpRew}(s \models \diamond\text{tar}).$$

The expected rewards of every state can be obtained by solving a linear equation system which has been presented in Chapter 2. The value $f_1(s)$ provides information about the benefit of the state s itself as well as the “future” of s , i.e., the benefit of the states that are reachable via s . However, there is no information about the “past” of s , i.e., the probability to reach s . This can lead to selected states that are only reachable with low probability and therefore less beneficial for building a critical subsystem.

The function f_2 can be used to take the “future” as well as the “past” of a state into consideration. Let $\pi_s^{s'}$ denote a finite path from state $s \in S$ to state $s' \in S$ with maximal

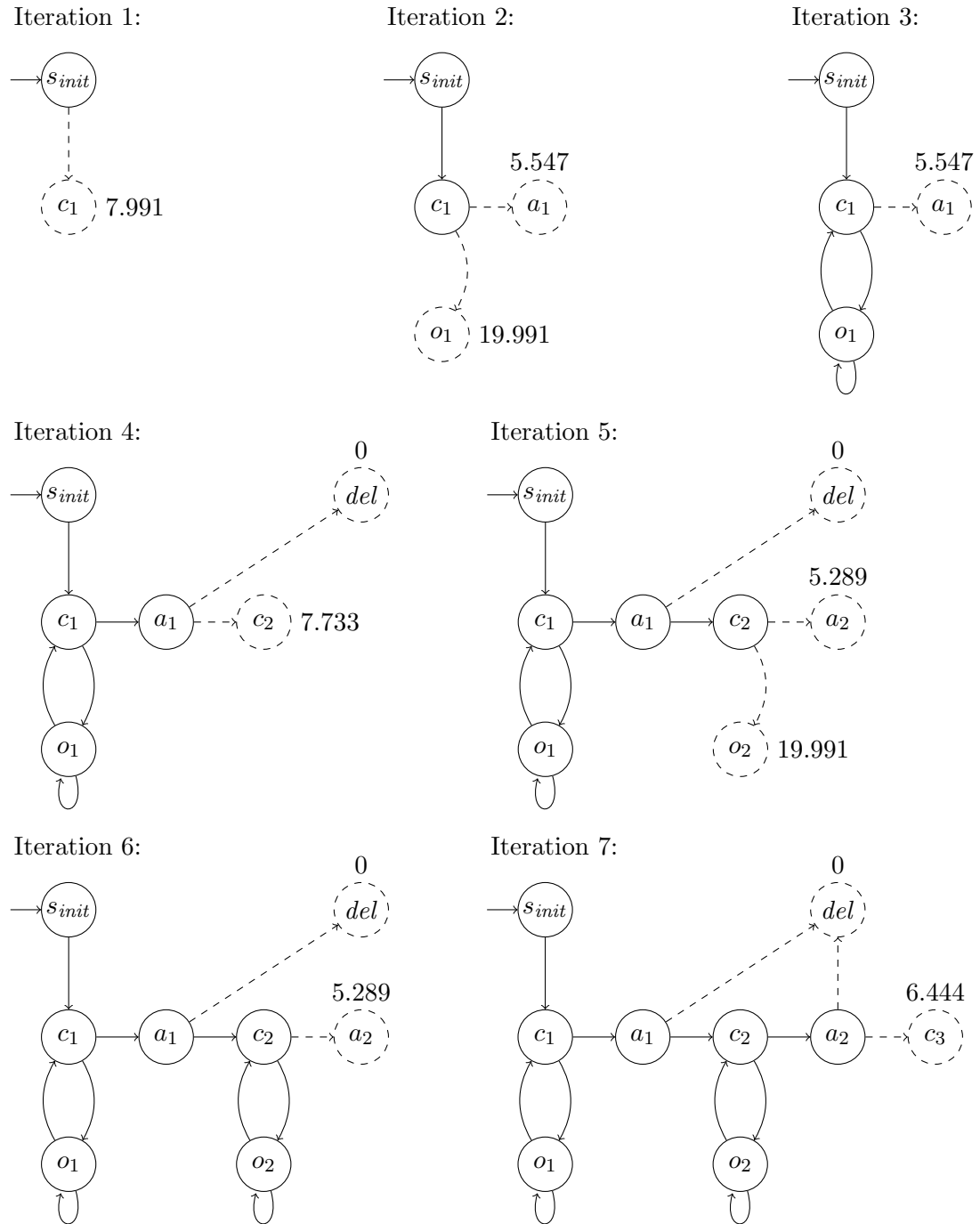


Figure 5.3: Illustration of the critical subsystem generation using the best-first search approach and the value function $f(s) = ExpRew(s \models \Diamond end)$.

probability, i.e.,

$$\mathbf{P}(\pi_s^{s'}) \geq \mathbf{P}(\hat{\pi}) \text{ for all paths } \hat{\pi} \text{ from } s \text{ to } s'.$$

The value function f_2 is given by

$$f_2(s) = \mathbf{P}(\pi_{s_{init}}^s) \cdot \text{ExpRew}(s \models \diamond \mathbf{tar}).$$

The probabilities $\mathbf{P}(\pi_{s_{init}}^s)$ for all states $s \in S$ can be obtained by using a variant of Algorithm 1. A drawback of this value function is that a state s with a high value of $\mathbf{P}(\pi_{s_{init}}^s)$ might be preferred over states that are equipped with high rewards but are not reachable via a single path with high probability. For instance, consider the MRM depicted in Figure 5.4. It holds that $\text{ExpRew}(s_i \models \diamond \mathbf{tar}) = 1$ for all $0 \leq i \leq 5$ and therefore it is easy to see that $f_2(s_i) > f_2(s_5)$ for all $0 \leq i \leq 4$. Hence, all states s_i for $0 \leq i \leq 4$ will be selected before s_5 , although s_5 is the only state that is equipped with positive reward. This problem can be avoided by using the function f_1 . For a state s with $\mathbf{tar} \notin L(s)$ it holds that $\text{rew}(s) > 0$ or there is a successor s' of s with $f_1(s) \leq f_1(s')$.

This also applies to the third value function f_3 which is given by

$$f_3(s) = \mathbf{P}(\pi_{s_{init}}^s) \cdot \max_{s' \in S} (\mathbf{P}(\pi_s^{s'}) \cdot \text{rew}(s')).$$

The values $\max_{s' \in S} (\mathbf{P}(\pi_s^{s'}) \cdot \text{rew}(s'))$ can be obtained by a variant of Algorithm 1 that uses backwards transitions. $f_3(s)$ also considers the “future” and the “past” of a state s . However, the computation relies on the approximation of the reachability of certain states in terms of the probability of a single path which might be misleading.

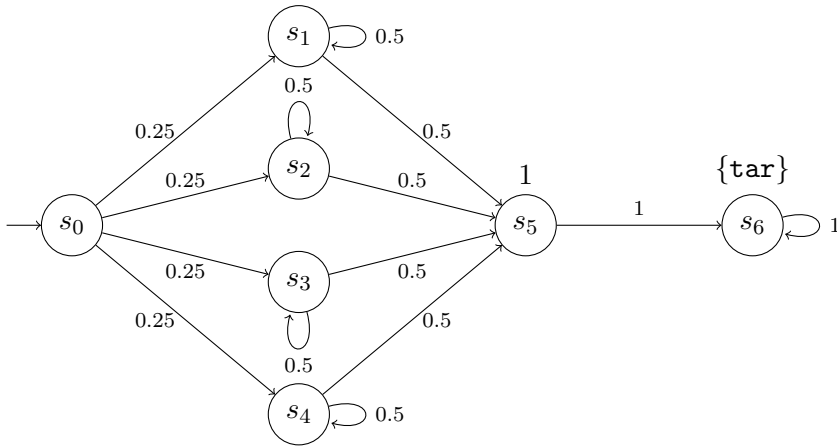


Figure 5.4: Example MRM to illustrate undesired behavior when using the function f_2 .

We have seen that no value function is clearly superior to all others. This also applies to the different modifications of the path search approach presented before. In the next chapter, we are going to see whether these observations also occur in practical tests.

Chapter 6

Experimental Results

The algorithms described in Chapter 5 have been implemented into the framework of StoRM which is a not yet released successor of MRMC [15], written in C++. We want to evaluate combinations of different approaches, models and properties. For this purpose, we measure the number of states of the generated subsystems as well as the run-time of the computation, excluding the time required to parse the model. All experiments have been performed on a 2.66 Ghz CPU with 6 GB RAM.

6.1 Models and Properties for the Experiments

We briefly introduce the models and properties that are used for our experiments. The models and most of the properties originate from the PRISM benchmark suite [16]. Furthermore, the PRISM tool [17] has been used to generate DTMCS and MRMs in an explicit representation which can be used as an input for our implementation.

Crowds protocol (crowds) [18]. The aim of the crowds protocol by Reiter and Rubin is to hide the identity of the sender of a message by randomly routing the message within a group of other stations (the crowd members). A fraction of the crowd members is considered to be bad, i.e., they want to collect information to reveal the identity of the sender. We assume that there are N good and $N \cdot 0.2$ bad crowd members. The time a single crowd member requires to forward or deliver a message is assumed to vary between 1 and 5 time units, depending on the specific crowd member. We want to consider the expected time required for K protocol runs (i.e., for delivering K messages). We are also interested in the probability that, within K protocol runs, a bad crowd member observes the originator of the message more than once.

Self-stabilization protocol (herman) [9]. Consider a ring of N identical processes. A configuration is called stable if there is exactly one designated process. The purpose of the self-stabilization protocol by Herman is to transform the system from an arbitrary configuration into a stable one. We are interested in the expected number of steps until the system reaches a stable configuration. For the experiments, an initial configuration is chosen in advance such that the expected number of steps is maximal.

Synchronous leader election protocol (leader_sync) [10]. Consider again a ring of N processes. The leader election protocol by Itai and Rodeh can be used to determine a designated leader of these processes. To attempt the election of a leader, every process has to choose a random id from $\{1, \dots, K\}$. The ids are exchanged among the processes. If there is a unique id, the process with the maximum unique id will become the leader. Otherwise, a new attempt is started by randomly choosing new ids. We are going to consider the probability that a leader will eventually be elected as well as the expected number of required attempts until a leader is elected.

Contract signing protocol (egl) [7]. In order to fairly exchange commitments to a contract, two parties A and B can adhere to the contract signing protocol of Even et al. It is assumed that both parties have N pairs of secrets of length L which will be exchanged. A party has committed to a contract whenever both secrets of one of its pairs are known by the other party. We are interested in the probability that B knows both secrets of its counterpart while A does not. Furthermore, we have a look at the expected number of messages that A needs to receive in order to know a pair of B where only the messages after B knows a pair of A are considered.

6.2 Results for Expected Reward Properties

Let us first focus on the generation of critical subsystems for expected reward properties. For the path search approach described in Section 5.1, we run experiments for some selected combinations of the presented value functions \mathbf{V}_1 , \mathbf{V}_2 , and \mathbf{V}_3 as well as different modifications. The tested modifications are depicted by

- none, if there is no modification,
- v if the value of a path $s_0 \dots s_n$ is weighted with $v(s_0)$, and
- vwR if the value of a path $s_0 \dots s_n$ is weighted with $v(s_0)$ and $w(s_n)$ and the states with positive rewards are used as target states.

For the best-first search approach from Section 5.2, we test the three described functions f_1 , f_2 , and f_3 . For all experiments, the subsystem has only been checked to be critical (i.e., the subsystem is build and model checking is performed) if at least $|S| \cdot 0.0001$ additional states have been selected since the last check (see Section 5.1.3).

model	$N(-K)$ $ S $	λ E.Rew. (total)	Path Search						Best-first Search			
			V_1		V_2			V_3	f_1	f_2	f_3	
			none	v	none	v	vwR	vwR				
crowds	5-3	13.4892	S'	166	97	109	89	117	115	136	111	123
	1198	26.9784	⌚	0.03	0.01	0.02	0.02	0.03	0.03	0.04	0.02	0.03
	5-6	26.9784	S'	1372	881	741	698	753	826	753	798	887
	18817	53.9568	⌚	2.50	1.39	1.25	1.11	2.14	2.05	0.99	1.18	1.34
		48.5612	S'	18015	6296	13465	4726	10212	7504	10670	8439	7638
		53.9568	⌚	135.21	23.75	83.36	15.14	82.66	49.68	118.19	84.62	69.70
	10-6	26.9784	S'	23327	2947	4306	2448	3763	4301	3781	4376	4376
	352535	53.9568	⌚	805.95	96.88	144.86	80.36	237.13	210.63	2.47	3.02	3.16
	15-6	26.9784	S'	TO	6210	TO	5209	TO	TO	10579	11317	14515
	2464168	53.9568	⌚	TO	2035.2	TO	1586.0	TO	TO	10.40	11.54	14.22
	48.5612	S'	TO	TO	TO	TO	TO	TO	TO	425089	484621	
	53.9568	⌚	TO	TO	TO	TO	TO	TO	TO	1271.1	1536.8	
20-6	26.9784	S'	TO	TO	TO	TO	TO	TO	23387	23387	31891	
10633591	53.9568	⌚	TO	TO	TO	TO	TO	TO	385.63	427.03	419.92	
herman	9	6.0000	S'	177	132	86	87	75	98	87	102	115
	512	12.0000	⌚	0.06	0.04	0.02	0.02	0.02	0.03	0.02	0.02	0.02
	13	12.3077	S'	571	396	234	233	232	350	233	337	402
	8192	24.6154	⌚	2.06	1.71	1.54	1.55	1.55	1.70	1.45	1.54	1.66
15	16.6667	S'	875	609	363	366	364	580	382	541	631	
32768	33.3333	⌚	19.92	19.66	19.53	19.33	19.49	19.84	18.83	19.13	19.54	
leader_sync	4-8	1.0224	S'	12087	12087	278	278	329	329	303	303	303
	12400	1.0449	⌚	31.18	31.16	0.23	0.23	0.23	0.21	0.24	0.24	0.24
	6-6	1.0290	S'	227823	227823	6610	6610	6798	6798	6786	6786	6786
	234210	1.0580	⌚	2630.7	2629.8	55.90	55.94	56.07	45.10	4.31	4.33	4.35
8-4	1.1577	S'	TO	TO	62523	62523	62781	62781	62699	62699	62699	
460239	1.3154	⌚	TO	TO	836.48	837.69	839.56	699.42	70.65	70.66	70.74	
egl	5-2	0.5757	S'	4717	13009	4281	4290	2482	5521	3815	3560	2849
	33790	1.1514	⌚	1.19	5.27	1.06	1.06	3.55	3.94	4.35	3.77	2.80
	5-8	1.0298	S'	66178	64367	4423	4423	4443	4510	19988	4725	4925
	156670	2.0596	⌚	47.94	45.23	1.40	1.39	29.27	26.36	35.41	3.21	3.89
7-8	0.6899	S'	TO	TO	TO	TO	TO	TO	424213	233613	134384	
3489790	1.3798	⌚	TO	TO	TO	TO	TO	TO	906.87	274.5	120.19	

Table 6.1: Results for the experiments regarding critical subsystem generation for expected reward properties of the form $\mathbb{E}_{<\lambda}(\diamond\text{tar})$. The size of the generated critical subsystems ($|S'|$) as well as the run-times (⌚) are depicted. All times are given in seconds. TO > 1h.

The results of the experiments regarding the presented expected reward properties are depicted in Table 6.1. Let us first compare the size (i.e., the number of states) of the resulting critical subsystems ($|S'|$). We can always find a value function and a modification such that the path search approach yields a subsystem that is smaller than the subsystems generated by the best-first search approach. On the other hand, the sizes of the subsystems generated by the best-first search approach have the same order of magnitude. Depending on the chosen value function and modification, the performance of the path search approach can even be worse. For most of the generated subsystems, the number of considered states is only a small fraction of the original state space. However, some of the tested value functions (and modifications) yield relatively large subsystems on certain inputs. This is also illustrated in Figure 6.1. A noticeable example for this observation is given by the `leader_sync` instances. When using the path search approach with the value function \mathbf{V}_1 , nearly every state is selected while other methods yield very small critical subsystems. This happens due to the fact that high probable paths to target states (i.e., states where a leader is elected) might be preferred over less probable paths that visit states with rewards (i.e., states where a new attempt to elect a leader is started). It can be observed that none of the tested approaches is clearly superior to all others, which is a typical characteristic for heuristic approaches. However, a rough trend between the different methods can be seen. Using \mathbf{V}_2 with modification v , for instance, yields relatively small critical subsystem for the tested models and properties while the results when using the function V_1 with no modification tend to use more states than other approaches.

For the path search approach, it is required to search a most valuable path in every iteration which can be very time-intensive. In fact, if we compare the run-times (⌚) of the path search approach and the best-first search approach depicted in Table 6.1, we can observe that the latter requires less time for large inputs. For small inputs, however, the run-times are almost similar. In Table 6.2, the time required for the different parts of the two approaches is depicted in more detail. For the path search approach, we can see that searching for most valuable paths takes a large part of the total run-time. The run-time of the best-first search, however, is dominated by the initialization time as well as the time required to build the current subsystem and check whether it is critical. For models with $|S| \cdot 0.0001 \approx 1$, one should note that the subsystem is checked less often when the path search approach is used since (at least) the states on the currently found path will be added between two checks. The implementation of the best-first search, on the other hand, allows to perform model checking after every newly selected state. For instance, see the results for crowds 5-6 in Table 6.2. We can see that the number of performed checks (`#check`) for the best-first search approach is almost similar to the number of selected states ($|S'|$). On small models, this can lead to (avoidable) higher run-times of the best-first search approach compared to the path search approach.

There is a relationship between the number of states in a generated critical subsystem and the time required to generate it: The larger a critical subsystem, the more states have to be searched in order to generate it and model checking on the currently found

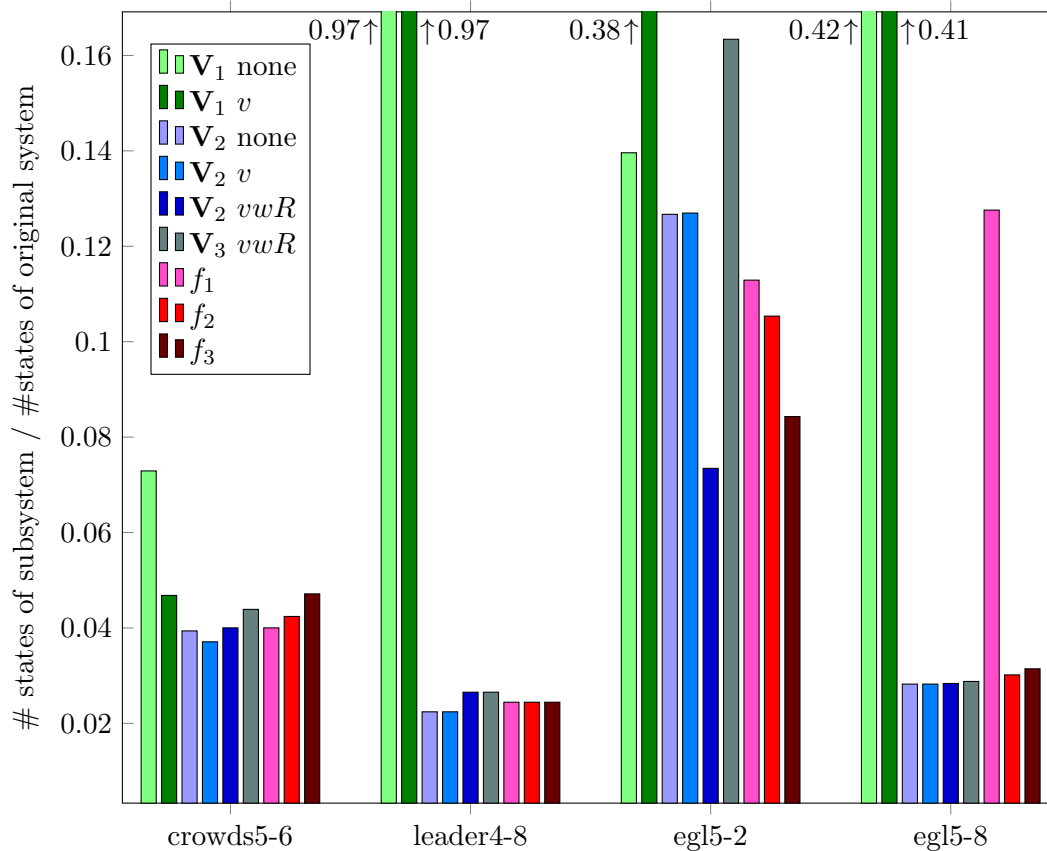


Figure 6.1: Illustration of the sizes of the critical subsystems generated by the different approaches.

N-K	Approach	$ S' $	\odot total	\odot initialization	\odot search	\odot check	#check	
crowds	5-6	V_2 v	698	1.11	0.03	0.75	0.33	212
		f_3	887	1.43	0.04	0.01	1.38	881
	10-6	V_2 v	2448	80.36	0.68	78.55	1.14	66
		f_3	4376	3.16	0.83	0.01	2.32	125
	20-6	f_3	31891	419.92	405.60	0.04	14.28	30

Table 6.2: Run-times of the different parts of the path search approach and the best-first search approach. The initialization includes model-checking of the original model as well as the computation of the values for V_2 and f_3 , respectively. The column #check denotes how often the subsystem has been checked during the generation. Different instances of the crowds protocol with bound $\lambda = 26.9784$ have been considered. All times are given in seconds

system will be performed more often. This leads to higher run-times. Therefore, we can observe that choosing a value function (and modification) such that the resulting critical subsystem is small also leads to a low run-time. In Figure 6.2, the number of states in the critical subsystem as well as the run-time is depicted in more detail. For the models crowds5-6 and crowds10-6, these two values have been plotted against different bounds λ . For crowds5-6, we can see that the plots for the number of states and the run-time look very similar which indicates the described relationship between these two values. Although the two plots for crowds10-6 look similar as well, the run-time for the depicted path search approach is considerably higher since searching a most valuable path is more time intensive on this larger models.

We can conclude that the path search approach can generate small critical subsystems fairly fast, providing that the size of the original model is small. For larger models, the run time of the path search approach heavily increases. The best-first search, on the other hand, trends to result in slightly larger critical subsystems, but has a more acceptable run-time on larger models. This observation is also illustrated in Figure 6.3. It can be assumed that the best-first search approach also shows good results on even larger models than depicted in Table 6.1. However, due to the yet unreleased status of SToRM, there is a high memory consumption while parsing and building the model. It should be noted that this is not a drawback of our presented approach as there is only a small need for memory.






model	$N(-K)$ $ S $ $ S_R $	λ E.Rew. (total)	Path Search							Best-first Search		
			V_1		V_2			V_3	f_1	f_2	f_3	
			none	v	none	v	vwR	vwR				
crowds	5-6	26.9784	$ S'_R $	198	222	146	163	280	189	280	179	228
	18817	53.9568		12.11	10.16	9.16	8.48	12.88	10.37	21.60	20.72	20.42
	4620	48.5612	$ S'_R $	3006	1431	3231	1132	2275	2255	2541	2170	2124
		53.9568		168.42	77.11	184.08	63.19	149.01	149.09	328.09	297.85	237.70
crowds	15-3	26.9784	$ S'_R $		1624		1328		2508	4614	2566	2702
	2464168	53.9568		TO		TO		TO				
	610470				2198.3		1721.8		2952.5	241.41	201.84	176.53
ecl	5-8	1.0298	$ S'_R $	1058	1047	1070	1070	1064	1056	1055	1062	1041
	156670	2.0596										
	2093			161.98	157.44	10.55	10.57	78.90	74.95	465.33	97.98	110.61

Table 6.3: Results for the experiments regarding critical reward subsystem generation for expected reward properties. The number of states in the generated subsystem that have positive rewards ($|S'_R|$) as well as the run-times () are depicted. All times are given in seconds. TO > 1 h.

Generation of critical reward subsystems. The implementation of the introduced algorithms has been modified to generate critical reward subsystems according

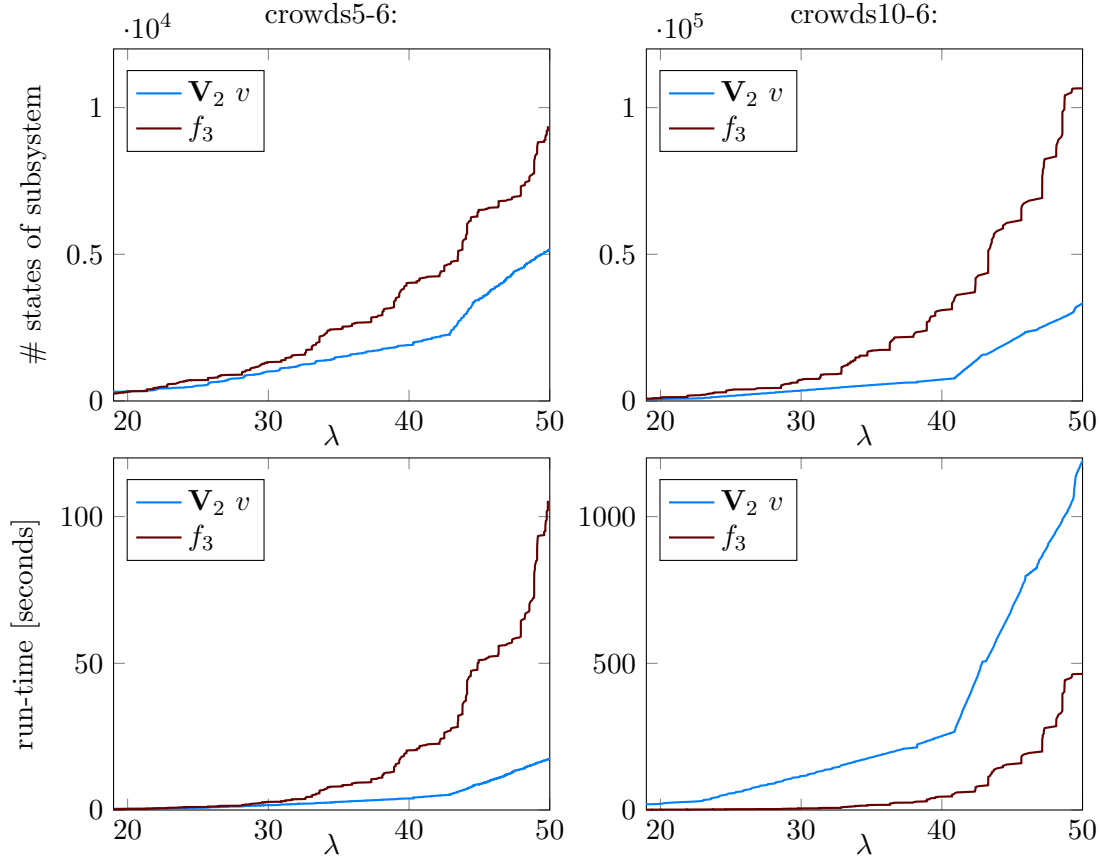


Figure 6.2: Resulting sizes of the generated critical subsystems (top) as well as the corresponding run-times (bottom), plotted against the reward bound λ . The plots refer to the crowds instances 5-6 (left, 18817 states) and 10-6 (right, 352535 states).

to Definition 4.3. Again, we run experiments for different inputs. The results are depicted in Table 6.3. The number of selected states with positive reward ($|S'_R|$ with $S'_R := \{s \in S \mid \text{rew}(s) > 0\}$) is lower compared to the sizes of the generated critical subsystems ($|S'|$) from Table 6.1. Analogous to the previously presented experiments, we can observe high discrepancies between the results when using certain value functions (and modifications). It should be noted that model checking is frequently performed on an MRM that considers all states. This occasionally causes higher run-times compared to the previously tested approaches, where model checking is (mostly) performed on a small subsystem.

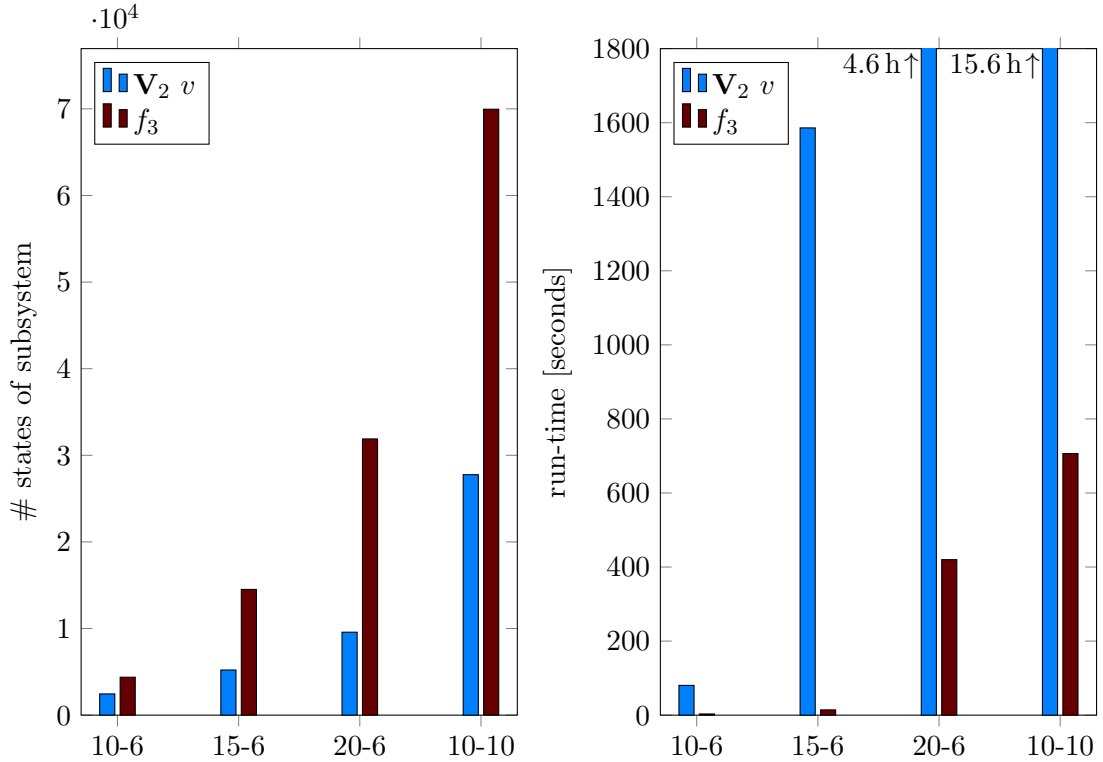


Figure 6.3: Comparison of the path search and the best-first search approach with regard to the resulting sizes of the critical subsystems (left) and the run-times (right). Different instances of the crowds protocol have been considered (for all tests we set $\lambda = 26.9784$): 10-6 (352535 states) 15-6 (2464168 states), 20-6 (10633591 states) and 10-10 (13201657 states).

6.3 Results for Reachability Properties

In Section 4.4, we showed how the problem of finding a critical subsystem of a DTMC for a reachability property could be reduced to the problem of finding a critical subsystem of an MRM for an expected reward property. We applied the given construction and used the best-first search approach to generate critical subsystems of the resulting MRM. This yields a new method to generate critical subsystems for reachability properties based on best-first search. The difference to the extended best-first search [2] is that the complete model is analyzed in advance resulting in a better informed search. Note that the presented path search approach could also be used for this purpose which would yield a method that is very similar to the already investigated local path search [11] and is therefore of little interest.

We compare the results of the best-first search approach using the three value functions f_1 , f_2 , and f_3 with the results of the local path search (local) implemented in COMICS

[12] and the extended best first search (XBF) implemented in DiPro [3]. We also consider the results of an implementation of the local path search in STORM. Here, we distinguish between two modifications – with and without weighting the probability of a path $s_0 \dots s_n$ with the probability of a most probable path from the initial state to s_0 . For the tests that use STORM, the subsystem is only checked if at least $|S| \cdot 0.0001$ states have been added since the last check.

Table 6.4 depicts the results of our experiments for reachability properties. Similar to the observations for expected reward properties, we can see that for most inputs the path search approaches yield smaller subsystems. However, the sizes of the subsystems generated by the best-first search approach are still comparable. For the `leader_sync` instances, this approach even yields smaller critical subsystems than the path search approaches. Considering the run-times of the various methods, the higher complexity of the path search approaches becomes apparent. For the large inputs that we have tested, the path search approaches can often not generate a critical subsystem without exceeding the time limit while our best first search approach yields a critical subsystems within a few seconds. It can also be observed that the run-times of the implementation of the extended best first search (XBF) exceed the run-times of our approach by orders of magnitude.

model	$N(-K)$ $ S $	λ Prob. (total)	COMICS	DiPro	StoRM		StoRM		
			local	XBF	Path Search none	v	Best-first Search f_1 f_2 f_3		
crowds	5-3	0.0692	$ S' $ 63 🕒	85	63	54	77	96	136
	1198	0.1383	🕒 0.04	0.56	0.01	0.01	0.02	0.02	0.03
	5-6	0.2135	$ S' $ 420 🕒	574	516	690	1136	849	731
	18817	0.4271	🕒 5.04	2.84	0.93	1.17	1.86	1.15	0.98
		0.3843	$ S' $ 2355 🕒	3447	5102	3439	3862	3909	3959
		0.4271	🕒 22.17	40.89	15.36	8.61	16.84	16.67	18.30
	10-6	0.1728	$ S' $ TO 🕒	4607	50884	3011	6512	3257	2627
	352535	0.3455	🕒 TO	76.41	1457.5	75.47	4.06	1.94	1.66
	15-6	0.1591	$ S' $ TO 🕒	12456	TO	10848	19928	7874	5906
	2464168	0.3182	🕒 TO	753.73	2801.3	2801.3	12.66	6.22	5.81
	0.2864	$ S' $ TO 🕒	TO	TO	TO	130136	108980	208610	
	0.3182	🕒 TO	TO	TO	TO	150.76	114.06	296.50	
20-6	0.1523	$ S' $ TO 🕒	23043	TO	TO	45711	14884	10632	
10633591	0.3046	🕒 TO	2229.04	TO	TO	30.64	16.70	17.41	
leader_sync	4-8	0.5000	$ S' $ 6521 🕒	6210	6344	6344	6335	6335	6193
	12400	1	🕒 15.83	20.82	6.90	6.89	22.16	21.44	19.82
	6-6	0.5000	$ S' $ TO 🕒	TO	121346	121346	120798	120798	117325
	234210	1	🕒 TO	TO	899.86	904.36	377.69	378.04	339.54
8-4	0.5000	$ S' $ TO 🕒	TO	271170	271170	261926	261926	230554	
460239	1	🕒 TO	TO	3323.1	3320.5	882.8	882.24	666.38	
egl	5-2	0.2578	$ S' $ 3438 🕒	3497	3586	3607	3690	3882	3881
	33790	0.5156	🕒 4.33	225.29	1.91	1.98	3.43	3.69	4.04
	5-8	0.2578	$ S' $ 19278 🕒	19401	19486	19447	19539	19794	19793
	156670	0.5156	🕒 79.69	1231.66	10.73	11.21	22.18	23.04	22.81
7-8	0.2520	$ S' $ TO 🕒	TO	TO	TO	421405	422148	422309	
3489790	0.5039	🕒 TO	TO	TO	TO	542.78	547.62	548.70	

Table 6.4: Results for the experiments regarding critical subsystem generation for reachability properties of the form $\mathbb{P}_{<\lambda}(\diamond \mathbf{tar})$. The size of the generated critical subsystems ($|S'|$) as well as the run-times (🕒) are depicted. All times are given in seconds. TO > 1 h.

Chapter 7

Conclusion and Future Work

In this theses, critical subsystems for expected reward properties have been presented to serve as counterexamples. The introduced definition of critical subsystems allows us to hide the parts of the system that do not need to be considered in order to understand why a given property is refuted. To efficiently generate small critical subsystems for expected reward properties, we discussed two heuristic approaches.

The path search approach is similar to the local path search, which generates critical subsystems for reachability properties and has been introduced in [11]. Value functions and other modifications have been suggested in order to take the rewards as well as the properties of a system into account. Experiments have shown that using this approach results in critical subsystems such that only a small fraction of the original system is considered.

The best-first search approach is related to the extended best-first search presented in [2]. However, information regarding the whole system is computed in advance and used to evaluate every state in terms of its benefit for a critical subsystem. Again, experiments yielded fairly small subsystems. The big advantage of this approach is the low run-time, even on bigger inputs. It can also be used to generate critical subsystems for reachability properties, yielding a new fast and effective method for this purpose.

Both presented approaches require to frequently perform model checking in order to check whether the currently found subsystem is critical. This takes a major part of the computation time, especially for the best-first search approach. Therefore, a good strategy to decide when model checking is performed can strongly improve the run-time of the generation. In this work, we tested a fairly simple strategy (only check every $|S| \cdot c$ newly selected states). There might be better methods for this purpose which can be analyzed in future work.

Although the presented heuristics show great promise, the size of the generated subsystems is not necessarily minimal. In [19], mixed integer linear programming has been used to show how minimal critical subsystems for reachability properties can be generated.

It can be assumed that minimal critical subsystems for expected reward properties can be generated by using a similar approach.

Another field of interest is the symbolic representation of DTMCs and MRMs. In this work, we assumed that the model is given in an explicit representation, meaning that sparse matrices are used to store the transition probabilities. For a symbolic representation (multi-terminal) binary decision diagrams are used to store the data of a DTMC or an MRM. Symbolic representations are less memory intensive and certain operations can be performed more efficient. In [13] and [14], the generation of critical subsystems for reachability properties by using symbolic representation of DTMCs has been presented. For the examined reachability properties, it has been shown that critical subsystems of very large DTMCs can be generated with comparably low time consumption. Hence, it is very promising to adapt the approaches presented in this thesis for symbolic representations.

References

- [1] Abraham E, Becker B, Dehnert C, Jansen N, Katoen JP, Wimmer R (2014) Counterexample generation for discrete-time Markov models: An introductory survey. In: Proc. of SFM, Springer, LNCS, vol 8483, pp 65–121
- [2] Aljazzar H, Leue S (2010) Directed explicit state-space search in the generation of counterexamples for stochastic model checking. *IEEE Trans on Software Engineering* 36(1):37–60
- [3] Aljazzar H, Leitner-Fischer F, Leue S, Simeonov D (2011) DiPro – A tool for probabilistic counterexample generation. In: Proc. of SPIN, Springer, LNCS, vol 6823, pp 183–187
- [4] Baier C, Katoen JP (2008) Principles of Model Checking. The MIT Press
- [5] Clarke EM (2008) The birth of model checking. In: 25 Years of Model Checking – History, Achievements, Perspectives, LNCS, vol 5000, Springer, pp 1–26
- [6] Dijkstra EW (1959) A note on two problems in connexion with graphs. *Numerische Mathematik* 1:269–271
- [7] Even S, Goldreich O, Lempel A (1985) A randomized protocol for signing contracts. *Communications of the ACM* 28(6):637–647
- [8] Han T, Katoen JP (2007) Counterexamples in probabilistic model checking. In: Proc. of TACAS, Springer, LNCS, vol 4424, pp 72–86
- [9] Herman T (1990) Probabilistic self-stabilization. *Information Processing Letters* 35(2):63–67
- [10] Itai A, Rodeh M (1990) Symmetry breaking in distributed networks. *Information and Computation* 88(1):60–87
- [11] Jansen N, Abraham E, Katelaan J, Wimmer R, Katoen JP, Becker B (2011) Hierarchical counterexamples for discrete-time Markov chains. In: Proc. of ATVA, Springer, LNCS, vol 6996, pp 443–452
- [12] Jansen N, Abraham E, Volk M, Wimmer R, Katoen JP, Becker B (2012) The

- COMICS tool – Computing minimal counterexamples for DTMCs. In: Proc. of ATVA, Springer, LNCS, vol 7561, pp 349–353
- [13] Jansen N, Abraham E, Zajzon B, Wimmer R, Schuster J, Katoen JP, Becker B (2012) Symbolic counterexample generation for discrete-time Markov chains. In: Proc. of FACS, Springer, LNCS, vol 7684, pp 134–151
- [14] Jansen N, Wimmer R, Abraham E, Zajzon B, Katoen JP, Becker B, Schuster J (2014) Symbolic counterexample generation for large discrete-time Markov chains. Science of Computer Programming 91, Part A(0):90–114
- [15] Katoen JP, Zapreev IS, Hahn EM, Hermanns H, Jansen DN (2011) The ins and outs of the probabilistic model checker MRMC. Performance Evaluation 68(2):90–104
- [16] Kwiatkowska M, Norman G, Parker D (2012) The PRISM benchmark suite. In: Proc. 9th International Conference on Quantitative Evaluation of Systems (QEST’12), IEEE CS Press, pp 203–204
- [17] Kwiatkowska MZ, Norman G, Parker D (2011) Prism 4.0: Verification of probabilistic real-time systems. In: Proc. of CAV, Springer, LNCS, vol 6806, pp 585–591
- [18] Reiter MK, Rubin AD (1998) Crowds: Anonymity for web transactions. ACM Transactions on Information and System Security 1(1):66–92
- [19] Wimmer R, Jansen N, Abraham E, Katoen JP, Becker B (2012) Minimal critical subsystems for discrete-time Markov models. In: Proc. of TACAS, Springer, LNCS, vol 7214, pp 299–314