Dual-Priced Modal Transition Systems with Time Durations

Nikola Beneš[2] Jan Křetínský[2,3] Kim G. Larsen[1]
Mikael H. Møller[1] Jiří Srba[1]

[1] Aalborg University, Denmark
[2] Masaryk University, Czech Republic
[3] Technical University München, Germany

Abstract. Modal transition systems are a well-established specification formalism for a high-level modelling of component-based software systems. We present a novel extension of the formalism called modal transition systems with durations where time durations are modelled as controllable or uncontrollable intervals. We further equip the model with two kinds of quantitative aspects: each action has its own running cost per time unit, and actions may require several hardware components of different costs. We ask the question, given a fixed budget for the hardware components, what is the implementation with the cheapest long-run average reward. We give an algorithm for computing such optimal implementations via a reduction to a new extension of mean payoff games with time durations and analyse the complexity of the algorithm.

1 Introduction and Motivating Example

Modal Transition Systems (MTS) is a specification formalism [14, 2] that aims at providing a flexible and easy-to-use compositional development methodology for reactive systems. The formalism can be viewed as a fragment of a temporal logic [1, 7] that at the same time offers a behavioural compositional semantics with an intuitive notion of process refinement. The formalism of MTS is essentially a labelled transition system that distinguishes two types of labelled transitions: *must* transitions which are required in any refinement of the system, and *may* transitions that are allowed to appear in a refined system but are not required. The refinement of an MTS now essentially consists of iteratively resolving the presence or absence of may transitions in the refined process.

In a recent line of work [13, 3], the MTS framework has been extended to allow for the specification of additional constraints on quantitative aspects (e.g. time, power or memory), which are highly relevant in the area of embedded systems. In this paper we continue the pursuit of quantitative extensions of MTS by presenting a novel extension of MTS with time durations being modelled as controllable or uncontrollable intervals. We further equip the model with two kinds of quantitative aspects: each action has its own running cost per time unit, and actions may require several hardware components of different costs. Thus, we ask the question, given a fixed budget for the investment into the hardware components, what is the implementation with the cheapest long-run average reward.

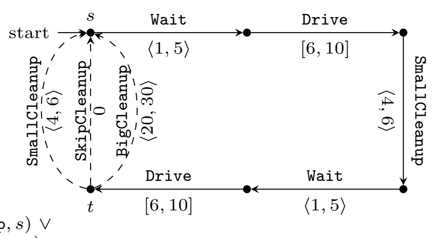
Before we give a formal definition of modal transition systems with durations (MTSD) and the dual-price scheme and provide algorithms for computing the optimal implementation, we present a small motivating example.

Consider the specification $\mathcal{S}$ for the work of a bus driver depicted in Figure 1a. He drives a shuttle bus between a hotel and the airport. First, he has to `Wait` for passengers at the hotel. This can take one to five minutes. Since this behaviour is required to be present in all the implementations of this specification it is drawn as a solid arrow and often called a *must* transition. Then the driver has to `Drive` the bus to the airport taking six to ten minutes, where he has to do a `SmallCleanup`, then `Wait` before he can `Drive` the bus back to the hotel. When he returns he can do either a `SmallCleanup`, `BigCleanup` or `SkipCleanup` of the bus before he continues. Here we do not require a particular option be realised in the implementations, hence we only draw the transitions as dashed arrows. As these transitions may or may not be present in the implementations they are called *may* transitions. However, here the intention is to require at least one option be realised. Hence, we specify this using a propositional formula Φ assigned to the state t over its outgoing transitions as described in [5, 6]. After performing one of the actions, the driver starts over again. Note that next time, the choice in t may be different.

Observe that there are three types of durations on the transitions. Firstly, there are *controllable* intervals, written in angle brackets. The meaning of e.g. $\langle 1, 5\rangle$ is that in the implementation we can instruct the driver to wait for a fixed number of minutes in the range. Secondly, there are *uncontrollable* intervals, written in square brackets. E.g. $[6, 10]$ on the `Drive` transition means that in the implementation we cannot fix any particular time and the time can vary say depending on traffic and is chosen nondeterministically by the environment. Thirdly, the degenerated case of a single number, e.g. 0, denotes that the time taken is always constant and given by this number. In particular, a zero duration means that the transition happens instantly.

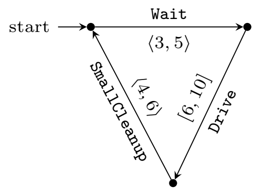
$\mathcal{S}_1$ is another specification, which is a *refinement* of $\mathcal{S}$ where on the top we specify that the driver must do a `SmallCleanup` after each `Drive`. Note that the `Wait` interval has been restricted. $\mathcal{I}_1$ is an implementation of $\mathcal{S}_1$ (and actually also $\mathcal{S}$), where all controllable time intervals have already been restricted to final single values: the driver must `Wait` for 5 minutes and do the `SmallCleanup` for 6 minutes. Note that uncontrollable intervals remain unresolved in the implementations and the time is nondeterministically chosen by the environment each time the action is performed. This reflects the inherent uncontrollable uncertainty of the timing, e.g. of traffic.

$\mathcal{S}_2$ is yet another specification and again a refinement of $\mathcal{S}$, where the driver can do a `BigCleanup` in t and possibly there is also an alternative allowed here of a `SmallCleanup`. Notice that both `SmallCleanup` intervals have been restricted and changed to uncontrollable. This means that we give up the control over the duration of this action and if this transition is implemented its duration will be

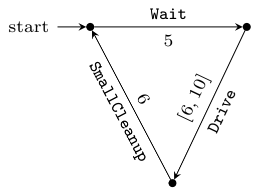


$\Phi(t) = (\texttt{BigCleanup}, s) \lor$
$(\texttt{SkipCleanup}, s) \lor$
$(\texttt{SmallCleanup}, s)$

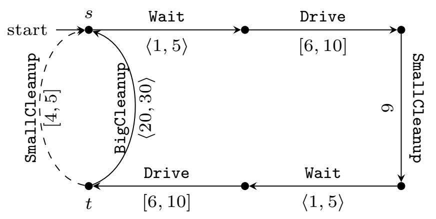
(a) Specification $\mathcal{S}$



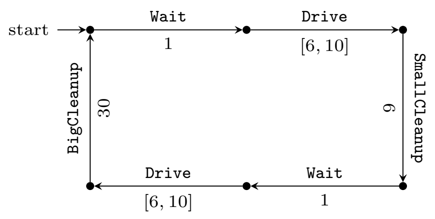
(b) Specification $\mathcal{S}_1$



(c) Implementation $\mathcal{I}_1$



(d) Specification $\mathcal{S}_2$



(e) Implementation $\mathcal{I}_2$

$a \in \Sigma$	$r(a)$
`Wait`	8
`Drive`	10
`SmallCleanup`	6
`BigCleanup`	7
`SkipCleanup`	0

$$H = \{\texttt{VacuumCleaner}, \texttt{Sponge}\}$$

$$\Psi(a) = \begin{cases} \texttt{VacuumCleaner} & \text{if } a = \texttt{BigCleanup} \\ \texttt{Sponge} \lor \texttt{VacuumCleaner} & \text{if } a = \texttt{SmallCleanup} \\ \mathit{true} & \text{otherwise} \end{cases}$$

$h \in H$	$i(h)$
`VacuumCleaner`	100
`Sponge`	5

(f) Price Scheme

Fig. 1: Example of Dual-Priced Modal Transition Systems with Time Durations

chosen every time nondeterministically in that range. $\mathcal{I}_2$ is then an implementation of $\mathcal{S}_2$ and $\mathcal{S}$, too.

Furthermore, we develop a way to model prices of the resources. Each action is assigned a *running price* it costs per time unit, e.g. `Drive` costs 10 each time unit it is being performed as can be seen in the left table of Figure 1f. In addition, in order to perform an action, some hardware may be needed, e.g. a `VacuumCleaner` for the `BigCleanup` and its price is 100 as can be seen on the right. This *investment price* is paid once only.

Let us now consider the problem of optimising the implementation, so that we spend the least possible amount of money (e.g. paid to the driver) per time unit while conforming to the specification $\mathcal{S}$. We call this problem *the cheapest implementation problem.* The optimal implementation is to buy a vacuum cleaner if one can afford an investment of 100 and do the `BigCleanup` every time as long as possible and `Wait` as shortly as possible. (Note that `BigCleanup` is more costly per time unit than `SmallCleanup` but lasts longer.) This is precisely implemented in $\mathcal{I}_2$ and the average cost per time unit is ≈ 7.97. If one cannot afford the vacuum cleaner but only a sponge, the optimal worst case long run average is then a bit worse and is implemented by doing the `SmallCleanup` as long as possible and `Wait` now as *long* as possible. This is depicted in $\mathcal{I}_1$ and the respective average cost per time unit is ≈ 8.10. For further details, see Example 5.

The most related work is [10] where prices are introduced into a class of interface theories and long-run average objectives are discussed. Our work omits the issue of distinguishing input and output actions. Nevertheless, compared to [10], this paper deals with the time durations, the one-shot hardware investment and, most importantly, refinement of specifications. Further, timed automata have also been extended with prices [4] and the long-run average reward has been computed in [8]. However, priced timed automata lack the hardware and any notion of refinement, too.

The paper is organized as follows. We introduce the MTS with the time durations in Section 2 and the dual-price scheme together with the problem of the cheapest implementation in Section 3. Section 4 presents the main results on the complexity of this problem. First, we state the complexity of this problem in general and in an important special case and prove the hardness part. The algorithms proving the complexity upper bounds are presented only after introducing an extension of mean payoff games with time durations. These are needed to establish the results but are also interesting on their own as discussed in Section 4.2. Due to space limitations, some of the proofs are in Appendix. We conclude and give some more account on related and future work in Section 5.

2 Modal Transition Systems with Durations

In order to define MTS with durations, we first introduce the notion of *controllable* and *uncontrollable* duration intervals. A controllable interval is a pair $\langle m, n\rangle$ where $m, n \in \mathbb{N}_0$ and $m \leq n$. Similarly, an uncontrollable interval is a pair $[m, n]$ where $m, n \in \mathbb{N}_0$ and $m \leq n$. We denote the set of all controllable intervals by $\mathfrak{I}_\mathsf{c}$, the set of all uncontrollable intervals by $\mathfrak{I}_\mathsf{u}$, and the set of all intervals by $\mathfrak{I} = \mathfrak{I}_\mathsf{c} \cup \mathfrak{I}_\mathsf{u}$. We also write only m to denote the singleton interval $[m, m]$. Singleton controllable intervals need not be handled separately as there is no semantic difference to the uncontrollable counterpart.

We can now formally define modal transition systems with durations. In what follows, $\mathcal{B}(X)$ denotes the set of propositional logic formulae over the set X of atomic propositions, where we assume the standard connectives $\wedge, \vee, \neg$.

Definition 1 (MTSD). *A* Modal Transition System with Durations *(MTSD) is a tuple $\mathcal{S} = (S, T, D, \Phi, s_0)$ where S is a set of* states *with designed* initial state s_0*,* $T \subseteq S \times \Sigma \times S$ *is a set of* transitions*,* $D : T \to \mathfrak{I}$ *is a* duration interval function*, and $\Phi : S \to \mathcal{B}(\Sigma \times S)$ is an* obligation function*. We assume that whenever the atomic proposition (a, t) occurs in the Boolean formula $\Phi(s)$ then also $(s, a, t) \in T$.*

We moreover require that there is no cycle of transitions that allows for zero accumulated duration, i.e. there is no path $s_1 a_1 s_2 a_2 \cdots s_n$ where $(s_i, a_i, s_{i+1}) \in T$ and $s_n = s_1$ such that for all i, the interval $D((s_i, a_i, s_{i+1}))$ is of the form either $\langle 0, n \rangle$ or $[0, n]$.

Note that instead of the basic may and must modalities known from the classical modal transition systems (see e.g. [2]), we use arbitrary boolean formulae over the outgoing transitions of each state in the system as introduced in [6]. This provides a higher generality as the formalism is capable to describe, apart from standard modal transition systems, also more expressive formalisms like disjunctive modal transition systems [15] and transition systems with obligations [5]. See [6] for a more thorough discussion of this formalism.

In the rest of the paper, we adapt the following convention when drawing MTSDs. Whenever a state s is connected with a solid arrow labelled by a to a state s', this means that in any satisfying assignment of the Boolean formula $\Phi(s)$, the atomic proposition (a, s') is always set to true (the transition *must* be present in any refinement of the system). Should this not be the case, we use a dashed arrow instead (meaning that the corresponding transition *may* be present in a refinement of the system but it can be also left out). For example in Figure 1a the solid edges correspond to an implicitly assumed $\Phi(s) = (a, s')$ where (s, a, s') is the (only) outgoing edge from s; in this case we do not explicitly write the obligation function. The three dashed transitions in the figure are optional, though at least one of them has to be preserved during any refinement (feature that can be modelled in for example the disjunctive MTS [15]).

Remark 2. The convention introduced above gives us the standard notation of modal transition systems (see e.g. [2]) under the restriction that the formulae $\Phi(s)$ in any state $s \in S$ have the form $(a_1, s_1) \wedge \ldots \wedge (a_n, s_n)$ where $(s, a_1, s_1), \ldots, (s, a_n, s_n) \in T$. The edges mentioned in such formulae are exactly all must transitions; may transitions are not listed in the formula and hence can be arbitrarily set to true or false.

Let by $T(s) = \{(a, t) \mid (s, a, t) \in T\}$ denote the set of all outgoing transitions from the state $s \in S$. A modal transition system with durations is called an *implementation* if $\Phi(s) = \bigwedge T(s)$ for all $s \in S$ (every allowed transition is also required), and $D(s, a, s') \in \mathcal{I}_{\mathsf{u}}$ for all $(s, a, s') \in T$, i.e. all intervals are uncontrollable, often singletons. Figure 1c shows an example of an implementation, while Figure 1b is not yet an implementation as it sill contains the controllable intervals $\langle 3, 5 \rangle$ and $\langle 4, 6 \rangle$.

We now define a notion of modal refinement. In order to do that, we first need to define refinement of intervals as a binary relation $\leq \subseteq \mathfrak{I} \times \mathfrak{I}$ such that

– $\langle m', n' \rangle \leq \langle m, n \rangle$ whenever $m' \geq m$ and $n' \leq n$, and
– $[m', n'] \leq \langle m, n \rangle$ whenever $m' \geq m$ and $n' \leq n$.

Thus the controllable intervals can be refined by narrowing the interval, at most until its a singleton interval, or until it is changed to an uncontrollable interval. Let us denote the collection of all possible sets of outgoing transitions from a state s by $\mathrm{Tran}(s) := \{E \subseteq T(s) \mid E \models \Phi(s)\}$.

Definition 3 (Modal Refinement). *Let $\mathcal{S}_1 = (S_1, T_1, D_1, \Phi_1, s_1)$ and $\mathcal{S}_2 = (S_2, T_2, D_2, \Phi_2, s_2)$ be two timed MTSs. A binary relation $R \subseteq S_1 \times S_2$ is a* modal refinement *if for every $(s,t) \in R$ the following holds:*

$$\begin{aligned}
&\forall M \in \mathrm{Tran}(s) : \exists N \in \mathrm{Tran}(t) : \\
&\qquad \forall (a, s') \in M : \exists (a, t') \in N : D_1(s, a, s') \leq D_2(t, a, t') \ \wedge \ (s', t') \in R \ \textit{and} \\
&\qquad \forall (a, t') \in N : \exists (a, s') \in M : D_1(s, a, s') \leq D_2(t, a, t') \ \wedge \ (s', t') \in R \ .
\end{aligned}$$

We say that $s \in S_1$ modally refines $s' \in S_2$, denoted by $s \leq_{\mathrm{m}} s'$, if there exists a modal refinement R such that $(s, s') \in R$. We also write $\mathcal{S}_1 \leq_{\mathrm{m}} \mathcal{S}_2$ if $s_1 \leq_{\mathrm{m}} s_2$.

Intuitively, the pair (s,t) can be in the relation R if for any satisfiable instantiation of outgoing edges from s there is a satisfiable instantiation of outgoing edges from t so that they can be mutually matched, possibly with s having more refined intervals, and the resulting states are again in the relation R.

Observe that in our running example the following systems are in modal refinement: $\mathcal{I}_1 \leq_{\mathrm{m}} \mathcal{S}_1 \leq_{\mathrm{m}} \mathcal{S}$ and thus also $\mathcal{I}_1 \leq_{\mathrm{m}} \mathcal{S}$, and similarly $\mathcal{I}_2 \leq_{\mathrm{m}} \mathcal{S}_2 \leq_{\mathrm{m}} \mathcal{S}$ and thus also $\mathcal{I}_2 \leq_{\mathrm{m}} \mathcal{S}$.

The reader can easily verify that on the standard modal transition systems (see Remark 2) the modal refinement relation corresponds to the classical modal refinement as introduced in [14].

3 Dual-Price Scheme

In this section, we formally introduce a dual-price scheme on top of MTSD in order to model the *investment cost* (cost of hardware necessary to perform the implemented actions) and the *running cost* (weighted long-run average of running costs of actions). We therefore consider only deadlock-free implementations (every state has at least one outgoing transition) so that the long-run average reward is well defined.

Definition 4 (Dual-Price Scheme). *A* dual-price scheme *over an alphabet Σ is a tuple $\mathcal{P} = (r, H, \Psi, i)$ where*

– *$r : \Sigma \to \mathbb{Z}$ is a* running cost *function of actions per time unit,*
– *H is a finite set of available* hardware*,*
– *$\Psi : \Sigma \to \mathcal{B}(H)$ is a* hardware requirement *function, and*
– *$i : H \to \mathbb{N}_0$ is a hardware* investment cost *function.*

Hence every action is assigned its unit cost and every action can have different hardware requirements (specified as a Boolean combination of hardware components) on which it can be executed. This allows for much more variability than a possible alternative of a simple investment cost $\Sigma \to \mathbb{N}_0$. Further, observe that the running cost may be negative, meaning that execution of such an action actually earns rather than spends money. This is then suitable for modelling e.g. vending machines.

Let $\mathcal{I}$ be an implementation with an initial state s_0. A set $G \subseteq H$ of hardware is *sufficient* for an implementation $\mathcal{I}$, written $G \models \mathcal{I}$, if $G \models \Psi(a)$ for every action a reachable from s_0. The *investment cost* of $\mathcal{I}$ is then defined as

$$\mathsf{ic}(\mathcal{I}) = \min_{G \models \mathcal{I}} \sum_{g \in G} i(g) \ .$$

Further, a *run* of $\mathcal{I}$ is an infinite sequence $s_0 a_0 t_0 s_1 a_1 t_1 \cdots$ with $(s_i, a_i, s_{i+1}) \in T$ and $t_i \in D(s_i, a_i, s_{i+1})$. Hence, in such a run, a concrete time duration in each uncontrollable interval is selected. We denote the set of all runs of $\mathcal{I}$ by $\mathcal{R}(\mathcal{I})$. The *running cost* of an implementation $\mathcal{I}$ is the worst-case long-run average

$$\mathsf{rc}(\mathcal{I}) = \sup_{s_0 a_0 t_0 s_1 a_1 t_1 \cdots \in \mathcal{R}(\mathcal{I})} \limsup_{n \to \infty} \frac{\sum_{i=0}^{n} r(a_i) \cdot t_i}{\sum_{i=0}^{n} t_i} \ .$$

Our *cheapest-implementation problem* is now defined as follows: given an MTSD specification $\mathcal{S}$ together with a dual-price scheme over the same alphabet, and given an upper-bound max_{ic} for the investment cost, find an implementation $\mathcal{I}$ of $\mathcal{S}$ (i.e. $\mathcal{I} \leq_{\mathrm{m}} \mathcal{S}$) such that $\mathsf{ic}(\mathcal{I}) \leq max_{\mathsf{ic}}$ and for every implementation $\mathcal{I}'$ of $\mathcal{S}$ with $\mathsf{ic}(\mathcal{I}') \leq max_{\mathsf{ic}}$, we have $\mathsf{rc}(\mathcal{I}) \leq \mathsf{rc}(\mathcal{I}')$.

Further, we introduce the respective decision problem, the *implementation problem* as follows: given an MTSD specification $\mathcal{S}$ together with a dual-price scheme, and given an upper-bound max_{ic} for the investment cost and an upper bound max_{rc} on the running cost, decide whether there is an implementation $\mathcal{I}$ of $\mathcal{S}$ such that both $\mathsf{ic}(\mathcal{I}) \leq max_{\mathsf{ic}}$ and $\mathsf{rc}(\mathcal{I}) \leq max_{\mathsf{rc}}$.

Example 5. Figure 1f depicts a dual-price scheme over the same alphabet $\Sigma = \{\mathtt{Wait}, \mathtt{Drive}, \mathtt{SmallCleanup}, \mathtt{BigCleanup}, \mathtt{SkipCleanup}\}$ as of our motivating specification $\mathcal{S}$. The running cost of the implementation $\mathcal{I}_2$ is $(1 \cdot 8 + 10 \cdot 10 + 6 \cdot 6 + 1 \cdot 8 + 10 \cdot 10 + 30 \cdot 7)/(1 + 10 + 6 + 1 + 10 + 30) \approx 7.97$ as the maximum value is achieved when `Drive` (with running cost 10) takes 10 minutes. On the one hand, this is optimal for $\mathcal{S}$ and maximum investment cost at least 100. On the other hand, if the maximum investment cost is 99 or less then the optimal implementation is depicted in $\mathcal{I}_1$ and its cost is $(5 \cdot 8 + 10 \cdot 10 + 6 \cdot 5)/(5 + 10 + 6) \approx 8.10$.

Remark 6. Note that the definition of the dual-price scheme only relies on having durations on the labelled transition systems. Hence, one could easily apply this in various other settings like in the special case of traditional MTS (with may and must transitions instead of the obligation function) or in the more general case of parametric MTS (see [6]) when equipped with durations as described above.

4 Complexity Results

4.1 Overview and Hardness Results

In this section, we give an overview of the complexity of our problem both in general and in an important special case, where it is easier. We start with establishing the hardness results. The matching upper bounds and the outline of their proofs follow. The subsequent sections are devoted to the actual proofs of the upper bounds and the corresponding algorithms.

We start by observing that the implementation problem is NP-hard even if no hardware is involved.

Proposition 7. *The implementation problem is NP-hard even for the hardware requirement function Ψ that is constantly true for all actions.*

Proof. We shall reduce the satisfiability problem of Boolean formulae (SAT) to our problem. Let φ be a Boolean formula over the variables $x_1, \ldots, x_n$. We define a MTSD $\mathcal{S}$ over the set of actions $\Sigma = \{x_1, \ldots, x_n, *\}$ such that the running cost is $r(x_j) = 1$ for all $1 \leq j \leq n$ and $r(*) = 2$ and the duration of all actions is 1. The specification $\mathcal{S}$ has one state s and a self-loop under all elements of Σ with the obligation function $\Phi(s) = \varphi \vee (*, s)$. The reason for adding the action $*$ is to make sure that in case φ is not satisfiable then we can still have a deadlock-free, but more running-cost-expensive implementation. Now we set the hardware to $H = \emptyset$ and the hardware requirement function $\Psi(a)$ identically true for all $a \in \Sigma$. It is easy to observe that the formula φ is satisfiable iff $\mathcal{S}$ has an implementation $\mathcal{I}$ with $\mathsf{rc}(\mathcal{I}) \leq 1$ (and $\mathsf{ic}(\mathcal{I}) = 0$). $\square$

Note that in the proof we required Φ to be a general Boolean formula. If, for instance, we considered only Φ positive (i.e. only containing $\wedge$ and $\vee$ operators and not $\neg$), the hardness would not hold. Thus on the one hand, one source of hardness is the complexity of Φ. On the other hand, even if Φ corresponds to the simplest case of an implementation (Φ is a conjunction of atomic propositions), the problem remains hard due to the hardware.

Proposition 8. *The implementation problem is NP-hard even for a specification that is already an implementation.*

Proof. We reduce the NP-complete problem of vertex cover to our problem. Let (V, E) where $E \subseteq V \times V$ be a graph and $k \in \mathbb{N}$ be an integer. We ask whether there is a subset of vertices $V_k \subseteq V$ of cardinality k such that for every $(v_1, v_2) \in E$ at least $v_1 \in V_k$ or $v_2 \in V_k$. Let us construct an MTSD specification $\mathcal{S}$ with hardware $H = V$ and the investment function $i(v) = 1$ for all $v \in H$, such that $\mathcal{S}$ has only one state s and a self-loop under the action a that is required ($\Phi(s) = (a, s)$) and where the hardware requirement function is $\Psi(a) = \bigwedge_{(u,v) \in E} (u \vee v)$. There is now a vertex cover in (V, E) of size $\leq k$ iff $\mathcal{S}$ has an implementation $\mathcal{I}$ with $\mathsf{ic}(\mathcal{I}) \leq k$. Setting e.g. $D(s, a, s) = 1$ and the running cost $r(a) = 0$ establishes NP-hardness of the implementation problem where we ask for the existence of an implementation of $\mathcal{S}$ with maximum running cost 0 and maximum investment cost k. $\square$

Note that hardware requirement function Ψ from the proof was in conjunctive normal form. However, if one only allowed Ψ in disjunctive normal form (DNF), the stated hardness would not hold.

In the subsequent sections, we also obtain the following matching upper bound which yields the following theorem.

Theorem 9. *The implementation problem is NP-complete.*

Nevertheless, combining both restrictions mentioned above yields a simpler problem as stated in the following theorem.

Theorem 10. *The implementation problem with Φ positive and Ψ in DNF is polynomially equivalent to mean payoff games and thus is in NP $\cap$ co-NP and can be solved in pseudo-polynomial time.*

This special case is extremely useful, as Ψ in DNF specifies the possible minimal configurations one by one, which is very natural. Moreover, transitions are usually prohibited by absence of the transition, not by negations in the formula. Thus, this case also encompasses e.g. the disjunctive MTS or transition systems with obligations. Nevertheless, in the obligation function Φ, sometimes implication and exclusive-or may be of some use. Therefore, one would like to deal with them if they occur from time to time. In Section 4.4 we show that one can do that using an algorithm where all negations involved in Φ only cause a *local* blow-up.

The subsequent sections are devoted to proving Theorems 9 and 10. The algorithm to solve the implementation problem first reduces the dual-priced MTSD into a mean payoff game extended with time durations and then solves this game. This new extension of mean payoff games and an algorithm to solve them is presented in Section 4.2. The translation follows in Section 4.3. Since this translation is exponential in general, Section 4.4 then shows how to translate in polynomial time with only local exponential blow-ups where negations occur. Section 4.5 then concludes and establishes the complexity results stated in the theorems.

4.2 Weighted Mean Payoff Games

We extend the standard model of mean payoff games (MPG) [12] with time durations. Not only is this extension needed for our algorithm, but it is also useful for modelling by itself. Consider, for instance, energy consumption of 2kW for 10 hours and 10kW for 2 hour, both followed by 10 hours of inactivity. Obviously, although both consumptions are 20kWh per cycle, the average consumption differs: 1kW in the former case and 20/12kW in the latter case. Furthermore, we allow zero durations. These are essential for our algorithm, but can also model e.g. a discrete change of state of a machine.

Definition 11. *A* weighted mean payoff game *is $G = (V, V_{min}, V_{max}, E, r, d)$ where V is a set of vertices divided into V_{min} and V_{max}, $E \subseteq V \times V$ is a set of edges, $r : E \to \mathbb{Z}$ is a rate function, $d : E \to \mathbb{N}_0$ is a duration function.*

It is assumed that there are no deadlocks (vertices with out-degree 0) and that there are no zero-duration cycles. The game is played by two players, *min* and *max*. The play is an infinite path such that each player picks a successor in his/her vertices. The value of a play $v_0v_1v_2\cdots$ is defined as:

$$\nu(v_0v_1v_2\cdots) = \limsup_{n\to\infty} \frac{\sum_{i=0}^{n} r(v_i, v_{i+1}) \cdot d(v_i, v_{i+1})}{\sum_{i=0}^{n} d(v_i, v_{i+1})} \tag{$*$}$$

Player *min* tries to minimize this value, while *max* aims at the opposite. Let $v(s)$ denote the infimum of the values *min* can guarantee if the play begins in vertex s no matter what *max* does.

Note that the standard MPGs in which edges are assigned integer weights only may be seen as weighted MPGs with rate equal to weight and duration equal to 1 on all edges.

We now show how to solve weighted MPGs by reduction to standard MPGs. We first focus on the problem whether $v(s) \geq 0$ for a given vertex s. As the durations are nonnegative and there are no zero-duration cycles, the denominator of the fraction in $(*)$ is always positive. Therefore, the following holds for every play $v_0v_1v_2\dots$ and every n:

$$\frac{\sum_{i=0}^{n} r(v_i, v_{i+1}) \cdot d(v_i, v_{i+1})}{\sum_{i=0}^{n} d(v_i, v_{i+1})} \geq 0 \iff \frac{1}{n}\sum_{i=0}^{n} r(v_i, v_{i+1}) \cdot d(v_i, v_{i+1}) \geq 0$$

We may thus solve the question whether $v(s) \geq 0$ by transforming the weighted MPG into a standard MPG, leaving the set of vertices and edges the same and taking $w(u,v) = r(u,v)\cdot d(u,v)$ as the edge weight function. Although the value $v(s)$ may change in this reduction, its (non)negativeness does not.

Further, we may transform any problem of the form $v(s) \geq \lambda$ for any fixed constant λ into the above problem. Let us modify the weighted MPG as follows. Let $r'(u,v) = r(u,v) - \lambda$ and leave everything else the same. The value of a play $v_0v_1v_2\cdots$ is thus changed as follows:

$$\nu'(v_0v_1v_2\cdots) = \limsup_{n\to\infty} \frac{\sum_{i=0}^{n} (r(v_i, v_{i+1}) - \lambda) \cdot d(v_i, v_{i+1})}{\sum_{i=0}^{n} d(v_i, v_{i+1})} = \nu(v_0v_1v_2\cdots) - \lambda$$

It is now clear that $v(s) \geq \lambda$ in the original game if and only if $v'(s) \geq 0$ in the modified game.

Furthermore, there is a one-to-one correspondence between the strategies in the original weighted MPG and the constructed MPG. Due to the two equivalences above, this correspondence preserves optimality. Therefore, there are optimal positional strategies in weighted MPGs since the same holds for standard MPGs [12]. (A strategy is positional if its decision does not depend on the current history of the play but only on the current vertex, i.e. can be described as a function $V \to V$.)

4.3 Translating Dual-priced MTSD into Weighted MPG

We first focus on the implementation problem without solving the hardware first, i.e. let us for the moment assume that $H = \emptyset$. We show how the implementation problem may be solved by reduction to the weighted MPGs. The first translation we present is going to be very large, we shall, however, provide methods for making the construction smaller in the subsequent section.

Suppose we are given an MTSD $\mathcal{S} = (S, T, D, \Phi, s_0)$ and a dual-price scheme (r, H, Ψ, i). We assume that there is no state s with $\emptyset \in \mathrm{Tran}(s)$, i.e. no deadlocks are allowed. Let us define the following auxiliary vertices that will be used to simulate the more complicated transitions of MTSD in the simpler setting of weighted MPG.

$$\begin{aligned}
T_{\mathsf{u}} &= \{(s,a,t) \mid (s,a,t) \in T;\ D(s,a,t) \in \mathcal{I}_{\mathsf{u}}\}\\
T_{\mathsf{c}} &= \{(s,a,t) \mid (s,a,t) \in T;\ D(s,a,t) \in \mathcal{I}_{\mathsf{c}}\}\\
T_{*} &= \{(s,a,j,t) \mid (s,a,t) \in T;\ j \in D(s,a,t)\}
\end{aligned}$$

We construct the weighted mean-payoff game with $V_{min} = S \cup T_{\mathsf{c}} \cup T_{*}$, $V_{max} = 2^T \cup T_{\mathsf{u}}$ and E defined as follows:

$$\begin{aligned}
(s, X) \in E &\iff \exists V \in \mathrm{Tran}(s) : X = \{(s,a,t) \mid (a,t) \in V\}\\
(X, (s,a,t)) \in E &\iff (s,a,t) \in X\\
((s,a,t),(s,a,j,t)) \in E &\iff j \in D(s,a,t)\\
((s,a,j,t),t) \in E &\quad \text{(always)}
\end{aligned}$$

Further, $r((s,a,j,t),t) = r(a)$, $d((s,a,j,t),t) = j$ and $r(-,-) = d(-,-) = 0$ otherwise.

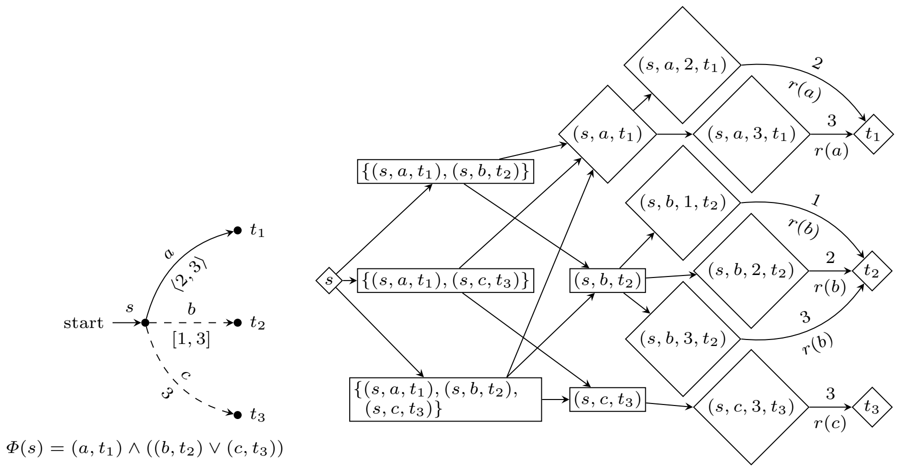
Example 12. In Figure 2 we have shown an example of how this translation to weighted MPG works. For simplicity we only translate a part of an MTSD $\mathcal{S}$ shown in Figure 2a. The resulting weighted MPG is shown in Figure 2b. The diamond shaped states belong to *min* and the squared states belong to *max*. In the vertex s, *min* chooses which outgoing transition are implemented. Only the choices satisfying $\Phi(s)$ are present in the game. Afterwards, *max* decides which transition to take. The chosen transition is then assigned by one of the players a time it is going to take.

Notice that (s, a, t_1) is the only transition controlled by *min*, because it has a controllable interval $\langle 2, 3 \rangle$. The remaining transitions with uncontrollable intervals are operated by *max* who chooses the time from these intervals. All the "auxiliary" transitions are displayed without any labels meaning their duration (and rate) is zero. Thus, only the transitions corresponding to "real" transitions in MTSDs are taken into account in the value of every play.

A strategy for *min* can now be translated into an implementation of the original MTSD in a straightforward way. The implemented transitions in s are given by $\sigma(s)$, similarly the durations of a transition (s, a, t) with a controllable interval is given by the third component of $\sigma((s, a, t))$. Details are given in Appendix.

4.4 Optimizations

We now simplify the construction. The first simplification is summarized by the following observation: The strategies of both players only need to choose the extremal points of the interval in vertices of the form (s, a, t).



(a) A part of an MTSD S (b) Resulting weighted MPG from translation of S

Fig. 2: Translating MTSD to weighted MPG.

Lemma 13. *There are optimal positional strategies for both* min *and* max *such that the choice in vertices of the form (s, a, t) is always one of the two extremal points of the interval $D(s, a, t)$.*

We may thus simplify the construction according to the previous lemma. The outgoing edges of states of the form (s, a, t) are defined as follows:

- $((s, a, t), (s, a, j, t)) \in E \iff j$ is an extremal point of $D(s, a, t)$

We thus reduce the number of outgoing edges for these vertices to at most two.

Secondly, we optimize the expansion of $\mathrm{Tran}(s)$. So far, we have built an exponentially larger weighted MPG graph as the size of $\mathrm{Tran}(s)$ is exponential in the out-degree of s. However, we can do much better, if we restrict ourselves to the class of MTSD where all $\Phi(s)$ are positive boolean formulae, i.e. the only connectives are $\wedge$ and $\vee$. Instead of enumerating all possible valuations, we can use the syntactic tree of the formula to build the weighted MPG.

Let $sf(\varphi)$ denote the set of all sub-formulae of φ (including φ). Let further $S_* = \{(s, \varphi) \mid s \in S;\ \varphi \in sf(\Phi(s))\}$. The weighted MPG is constructed with

- $V_{min} = \{(s, \varphi) \in S_* \mid \varphi = \varphi_1 \vee \varphi_2$ or $(\varphi = (a, t)$ and $D(s, a, t) \in \mathcal{I}_{\mathsf{c}})\} \cup T_*$
- $V_{max} = \{(s, \varphi) \in S_* \mid \varphi = \varphi_1 \wedge \varphi_2$ or $(\varphi = (a, t)$ and $D(s, a, t) \in \mathcal{I}_{\mathsf{u}})\}$
- E is defined as follows:

$$\begin{aligned}
((s, \varphi_1 \wedge \varphi_2), (s, \varphi_i)) \in E &\quad i \in \{1, 2\} \\
((s, \varphi_1 \vee \varphi_2), (s, \varphi_i)) \in E &\quad i \in \{1, 2\} \\
((s, (a, t)), (s, a, j, t)) \in E &\quad \iff j \text{ is an extremal point of } D(s, a, t) \\
((s, a, j, t), (t, \Phi(t))) \in E &\quad \text{(always)}
\end{aligned}$$

- $r((s, a, j, t), (t, \Phi(t))) = r(a)$ and $r(-, -) = 0$ otherwise
- $d((s, a, j, t), (t, \Phi(t))) = j$ and $d(-, -) = 0$ otherwise

Lemma 14. *Both optimizations are correct.*

Example 15. In Figure 3 we have shown the result of translating the part of an MTSD from Figure 2a. This weighted MPG is similar to the one in Figure 2b, but instead of having a vertex for each satisfying set of outgoing transitions we now have the syntactic tree of the obligation formula for each state. Further the vertex $(s, b, 2, t_2)$ is left out, due to Lemma 13. Note that the vertices $(t_1, \Phi(t_1))$, $(t_2, \Phi(t_2))$ and $(t_3, \Phi(t_3))$ are drawn as circles, because the player of these states depends on the obligation formula and the outgoing transitions.

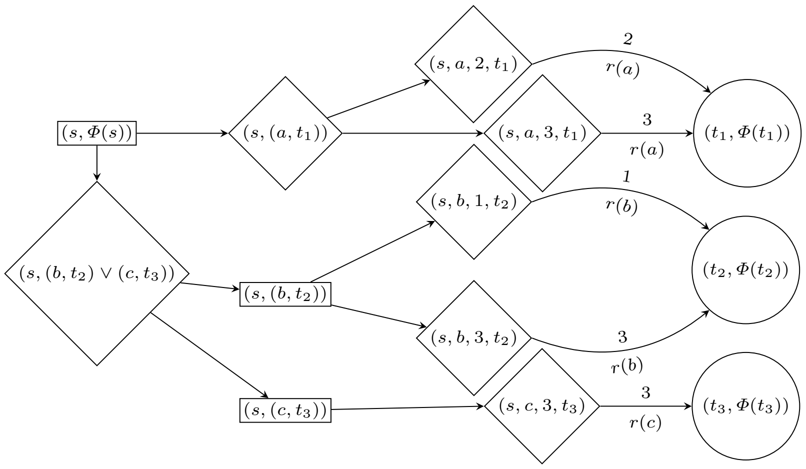


Fig. 3: Result of the improved translation of S in Figure 2a

Remark 16. Observe that one can perform this optimization even in the general case. Indeed, for those s where $\Phi(s)$ is positive we locally perform this transformation; for s with $\Phi(s)$ containing negations we stick to the original expansion. Thus, the exponential (in out-degree) blow-up occurs only locally.

4.5 Algorithm and its Complexity

The algorithm for our problem works as follows:

1. nondeterministically choose hardware with the total price at most max_{ic},
2. create the weighted MPG out of $\mathcal{S}$,
3. solve the weighted MPG using the reduction to MPG and any standard algorithm for MPG, which finds an optimal strategy for player *min* and computes the value $v(s_0)$,
4. transform the strategy to an implementation $\mathcal{I}$,
5. in the case of the cheapest-implementation problem return $\mathcal{I}$;
   in the case of the implementation (decision) problem return $v(s_0) \leq max_{\mathsf{rc}}$.

We can now prove the following complexity result, which also finishes the proof of Theorem 9. Note that the restricted case has a better complexity.

Proposition 17. *The implementation problem is in NP.*

Proof. We first nondeterministically guess the hardware assignment. Due to Section 4.3, we know that the desired implementation has the same states as the original MTSD and its transitions are a subset of the transitions of the original MTSD as the corresponding optimal strategies can be chosen positional. Due to the first optimization in Section 4.4, we also know that the durations can be chosen to be the extremal points of the intervals. Therefore, we can nondeterministically guess an optimal implementation and its durations and verify that it satisfies the desired price inequality. □

Proposition 18. *The implementation problem for MTSD with Φ positive and Ψ in DNF is in NP $\cap$ coNP and there is a pseudo-polynomial algorithm for this problem.*

Proof. With Ψ being in DNF, each clause yields a hardware configuration and we can check each configuration separately one by one. Further, by the first and second optimizations in Section 4.4, the MPG graph is of size $\mathcal{O}(|T| + |\Phi|)$. Therefore, we polynomially reduce the implementation problem to the problem of solving polynomially many (linearly many in $|\Psi|$) mean payoff games. The result follows by existence of pseudo-polynomial algorithms for MPGs [16]. □

Further, our problem is at least as hard as solving MPGs, which are clearly a special case of our problem. Hence, Theorem 10 follows.

5 Conclusion and Future Work

We have introduced a new extension of modal transition systems. The extension consists in introducing (1) variable time durations of actions and (2) a pricing of actions, where we combine one-shot investment price for the hardware and cost for running it per each time unit it is active. We believe that this formalism is appropriate to modelling many types of embedded systems, where safety comes along with economical requirements.

We have solved the problem of finding the cheapest implementation w.r.t. the running cost given a maximum hardware investment we can afford. The established complexity of the decision problem is summarized as follows.

Corollary 19. *The implementation problem is NP-complete except for the case with Φ positive and Ψ in disjunctive normal form when it is polynomially equivalent to mean payoff games and there exists a pseudo-polynomial algorithm solving the problem.*

In the most practical subclass (positive Φ with Ψ in DNF), the cheapest implementation can be constructed using the current algorithms for mean payoff games [16] in time $\mathcal{O}(|\Psi| \cdot (|T| + |\Phi|)^5 \cdot W)$, where W is the largest number such that $W = r(a) \cdot n$ where $n \in D(s, a, t)$.

As for the future work, apart from implementing the algorithm, one may consider two types of extensions. Firstly, one can extend the formalism to cover the distinction between input, output and internal actions as is usual in interface theories [10], and include even more time features, such as clocks in priced timed automata [4, 8], to get "Modal input/output priced timed automata". Secondly, one may extend the criteria for synthesis of the cheapest implementation by additionally requiring that the partial sums stay in bounds as in [9], or requiring satisfaction of a temporal property as suggested in [10, 11].

References

1. Aceto, L., Fábregas, I., de Frutos-Escrig, D., Ingólfsdóttir, A., Palomino, M.: Graphical representation of covariant-contravariant modal formulae. In: EXPRESS. EPTCS, vol. 64, pp. 1–15 (2011)
2. Antonik, A., Huth, M., Larsen, K.G., Nyman, U., Wasowski, A.: 20 years of modal and mixed specifications. Bulletin of the EATCS no. 95 pp. 94–129 (2008)
3. Bauer, S.S., Fahrenberg, U., Juhl, L., Larsen, K.G., Legay, A., Thrane, C.R.: Quantitative refinement for weighted modal transition systems. In: MFCS. LNCS, vol. 6907, pp. 60–71. Springer (2011)
4. Behrmann, G., Larsen, K.G., Rasmussen, J.I.: Priced timed automata: Algorithms and applications. In: FMCO. LNCS, vol. 3657, pp. 162–182. Springer (2004)
5. Beneš, N., Křetínský, J.: Process algebra for modal transition systemses. In: MEMICS. OASICS, vol. 16, pp. 9–18. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, Germany (2010)
6. Beneš, N., Křetínský, J., Larsen, K., Møller, M., Srba, J.: Parametric modal transition systems. In: Proceedings of ATVA'11. LNCS, vol. 6996, pp. 275–289. Springer-Verlag (2011)
7. Boudol, G., Larsen, K.G.: Graphical versus logical specifications. Theor. Comput. Sci. 106(1), 3–20 (1992)
8. Bouyer, P., Brinksma, E., Larsen, K.G.: Optimal infinite scheduling for multi-priced timed automata. Formal Methods in System Design 32(1), 3–23 (2008)
9. Bouyer, P., Fahrenberg, U., Larsen, K.G., Markey, N., Srba, J.: Infinite runs in weighted timed automata with energy constraints. In: FORMATS. LNCS, vol. 5215, pp. 33–47. Springer (2008)
10. Chakrabarti, A., de Alfaro, L., Henzinger, T.A., Stoelinga, M.: Resource interfaces. In: Alur, R., Lee, I. (eds.) EMSOFT. Lecture Notes in Computer Science, vol. 2855, pp. 117–133. Springer (2003)
11. Chatterjee, K., Doyen, L.: Energy parity games. In: Abramsky, S., Gavoille, C., Kirchner, C., auf der Heide, F.M., Spirakis, P.G. (eds.) ICALP (2). Lecture Notes in Computer Science, vol. 6199, pp. 599–610. Springer (2010)
12. Ehrenfeucht, A., Mycielski, J.: Positional strategies for mean payoff games. International Journal of Game Theory 8, 109–113 (1979), 10.1007/BF01768705
13. Juhl, L., Larsen, K.G., Srba, J.: Introducing modal transition systems with weight intervals. Journal of Logic and Algebraic programming (2011), to appear
14. Larsen, K.G., Thomsen, B.: A modal process logic. In: LICS. pp. 203–210. IEEE Computer Society (1988)
15. Larsen, K.G., Xinxin, L.: Equation solving using modal transition systems. In: LICS. pp. 108–117. IEEE Computer Society (1990)
16. Zwick, U., Paterson, M.: The complexity of mean payoff games on graphs. Theoretical Computer Science 158, 343–359 (1996)

A Proofs from Section 4.3 (Correctness of the Reduction)

We first show formally how a strategy for player *min* in the constructed weighted MPG may be translated into an implementation of the original MTSD. In Section 4.2, we have shown that there are optimal *positional* strategies. Hence, we may safely restrict this translation to positional strategies, and for sake of simplicity, we do so. Let σ be such a positional strategy. We build the implementation as $\mathcal{I} = (S', T', D', \Phi', s_{0\mathsf{I}})$ where

- $S' = \{s_\mathsf{I} \mid s \in S\}$
- $(s_\mathsf{I}, a, t_\mathsf{I}) \in T'$ if $(s, a, t) \in X$ where $\sigma(s) = X$
- $D'(s_\mathsf{I}, a, t_\mathsf{I}) = D(s, a, t)$ if $D(s, a, t) \in \mathcal{I}_\mathsf{u}$
- $D'(s_\mathsf{I}, a, t_\mathsf{I}) = j$ if $D(s, a, t) \in \mathcal{I}_\mathsf{c}$ and $\sigma((s, a, t)) = (s, a, j, t)$
- $\Phi'(s_\mathsf{I}) = \bigwedge_{(s_\mathsf{I}, a, t_\mathsf{I}) \in T'} (a, t_\mathsf{I})$

The set of states remains the same as in the original MTSD; we change every state s into s_I to be able to reason about the states of the original MTSD and the implementation separately. The following two lemmata state that the construction is sound.

Lemma 20. *The constructed implementation $\mathcal{I}$ is indeed an implementation of the original MTSD.*

Proof. We show that $R = \{(s_\mathsf{I}, s) \mid s \in S\}$ is a modal refinement relation. Let $(s_\mathsf{I}, s) \in R$ and let $M \in \mathrm{Tran}(s_\mathsf{I})$. Clearly $M = \{(a, t_\mathsf{I}) \mid (s_\mathsf{I}, a, t_\mathsf{I}) \in T'\}$. We take $N = \{(a, t) \mid (a, t_\mathsf{I}) \in M\}$. The fact that $N \in \mathrm{Tran}(s)$ is clear from the construction. We now need to show that $D'(s_\mathsf{I}, a, t_\mathsf{I}) \leq D(s, a, t)$. If $D(s, a, t) \in \mathcal{I}_\mathsf{u}$ then $D'(s_\mathsf{I}, a, t_\mathsf{I}) = D(s, a, t)$ and the statement holds. If $D(s, a, t) \in \mathcal{I}_\mathsf{c}$ then $D'(s_\mathsf{I}, a, t_\mathsf{I}) = j$ where $j \in D(s, a, t)$ due to the construction. Thus R is a modal refinement relation and $s_\mathsf{I} \leq_\mathrm{m} s$ for all $s \in S$, thus also $s_{0\mathsf{I}} \leq_\mathrm{m} s_0$ and therefore $\mathcal{I} \leq_\mathrm{m} \mathcal{S}$. □

Lemma 21. *We have $\mathsf{rc}(\mathcal{I}) \leq \lambda$ if and only if the strategy σ ensures a value at most λ.*

Proof. Every run of the implementation corresponds to a play (where player *min* plays according to strategy σ) and vice versa. Hence, the worst case of the long run average over all runs in $\mathcal{I}$ is the same as over all plays according to σ. □

We conclude the proof of correctness of the reduction by showing its completeness.

Lemma 22. *For every implementation $\mathcal{I}$ of $\mathcal{S}$, there exists a strategy σ for player* min *such that σ ensures value of at most $\mathsf{rc}(\mathcal{I})$.*

Proof. We show how to transform an arbitrary implementation $\mathcal{I}$ into a strategy $\sigma : V^* \to V$ (dependent on the whole prefix of a play where we currently are) for player *min* that guarantees the same or smaller value. (Although this is not a positional strategy, we know there is also a positional strategy ensuring the same or smaller value.) The idea of the construction is that for each history σ mimics the behaviour of the implementation at its respective state. In other words, the decision of σ in some history is based on mapping the history to the implementation and doing what the implementation does. However, there is a small catch: the implementation may implement more possible decisions (its branching may be even uncountable). Nonetheless, we may take any of the decisions, as the resulting strategy then corresponds to a pruning of the original implementation and the strategy's worst case long run average is thus either the same or even smaller.

Let i_0 be the initial state of $\mathcal{I}$ and s_0 the inital state of $\mathcal{S}$; we have $i_0 \leq_{\mathrm{m}} s_0$. The corresponding vertex of the constructed weighted MPG also bears the name s_0. We first define a mapping $\mu : V^* \to I^*$ from paths in the weighted MPG of the form $s_0\,(s_0, N_0)\,(s_0, a_0, s_1)\,(s_0, a_0, j_0, s_1)\,s_1\,\cdots s_n$ to sequences of states of $\mathcal{I}$ inductively as follows.

$$\begin{aligned}\mu(s_0) &= i_0\\ \mu(s_0 \cdots s_n\,(s_n, N_n)\,(s_n, a_n, s_{n+1})\,(s_n, a_n, j_n, s_{n+1})\,s_{n+1}) &= \mu(s_0 \cdots s_n) i_{n+1}\end{aligned}$$

where i_{n+1} is an arbitrary state satisfying $(i_n, a_n, i_{n+1}) \in T$ and $i_{n+1} \leq_{\mathrm{m}} s_{n+1}$. The image of μ now defines the desired pruning of $\mathcal{I}$ that is still an implementation of $\mathcal{S}$, has at most the same worst case long run average, and can now be canonically mapped to a deterministic strategy σ as follows.

Let $\pi = s_0\,(s_0, N_0)\,(s_0, a_0, s_1)\,(s_0, a_0, j_0, s_1)\,s_1\,\cdots x \in V^*$ be a prefix of a play with $x \in V_{min}$. Denote s_n the last element of $\mathcal{S}$ in π and $i_0 \cdots i_n = \mu(s_0 \cdots s_n)$. We define $\sigma(\pi)$ as follows.

- If $x \in S$, then $\pi = s_0 \cdots s_n$ and due to $i_n \leq_{\mathrm{m}} s_n$ there exists N from the definition of modal refinement. We set $\sigma_s(\pi) = (s_n, N)$.
- If $x \in T_{\mathsf{c}}$, then $\pi = s_0 \cdots s_n\,(s_n, N_n)\,(s_n, a_n, s_{n+1})$ and there exists a unique i_{n+1} such that $i_0 \cdots i_n i_{n+1} = \mu(s_0 \cdots s_n s_{n+1})$. Here we used the fact that μ does not depend on which vertices of the form (s, a, j, t) were visited. We set $\sigma(\pi) = (s_n, a_n, D(i_n, a_n, i_{n+1}), s_{n+1})$.
- If $x \in T_*$, then $x = (s_n, a_n, j_n, s_{n+1})$ and there exists only one outgoing edge. We set $\sigma(\pi) = s_{n+1}$.

Now for each play that player *min* plays according to σ, there is a run in the original implementation that has the same long-run average. Hence the supremum over all plays is at most the supremum over all runs. □

B Proofs from Section 4.4 (Correctness of the Optimizations)

Lemma 13. *There are optimal positional strategies for both* min *and* max *such that the choice in vertices of the form (s, a, t) is always one of the two extremal points of the interval $D(s, a, t)$.*

Proof. Let σ and ρ be any optimal positional strategies for *min* and *max*, respectively. We will transform them into σ' and ρ', optimal strategies where the choice in vertices of the form (s,a,t) is always one of the two extremal points of the interval $D(s,a,t)$. We show the transformation for *min*'s strategy σ. The case with ρ is symmetric. The proof is done by induction on the number of vertices of the form (s,a,t) from which σ chooses a nonextremal point of the interval $D(s,a,t)$.

If there are no such vertices, we are obviously done. Suppose further that there is at least one such vertex, say (s,a,t). We thus have $D(s,a,t) = \langle m,n \rangle$ and $\sigma((s,a,t)) = (s,a,j,t)$ with $m < j < n$.

We investigate three cases, depedning on the relationship of the rate $r(a)$ and the value $v(s)$.

- $r(a) = v(s)$. Then the duration of the transition (s,a,t) does not matter and we may freely change σ into σ' by defining $\sigma'((s,a,t)) = (s,a,n,t)$ and $v(s)$ remains the same.
- $r(a) > v(s)$. Clearly, changing σ into σ' by defining $\sigma'((s,a,t)) = (s,a,m,t)$ may only decrease $v(s)$ or not change it at all. (As σ is optimal strategy, $v(s)$ remains the same using σ'.)
- $r(a) < v(s)$. Using similar argument as in the previous case, we change σ into σ' by defining $\sigma'(s,a,t) = (s,a,n,t)$ and σ' remains optimal.

Repeated use of this argument concludes the proof. $\square$

Lemma 14. *Both optimizations are correct.*

Proof. A strategy for the player *min* of a weighted MPG from a translation utilizing the optimizations of Section 4.4 can be translated into an implementation of the original MTSD as follows. Let σ be the strategy. We first define an auxiliary function on the vertices of the MPG as follows:

$$\begin{aligned} f(s,(a,t)) &= \{(a,t)\} \\ f(s,\varphi \vee \psi) &= f(\sigma(s,\varphi \vee \psi)) \\ f(s,\varphi \wedge \psi) &= f(s,\varphi) \cup f(s,\psi) \end{aligned}$$

We then build the implementation as (S', T', D', Φ') where

- $S' = \{s_\mathsf{I} \mid s \in S\}$
- $(s_\mathsf{I}, a, t_\mathsf{I}) \in T'$ if $(a,t) \in f(s,\Phi(s))$
- $D'(s_\mathsf{I}, a, t_\mathsf{I}) = D(s,a,t)$ if $D(s,a,t) \in \mathfrak{I}_\mathsf{u}$
- $D'(s_\mathsf{I}, a, t_\mathsf{I}) = j$ if $D(s,a,t) \in \mathfrak{I}_\mathsf{c}$ and $\sigma((s,(a,t))) = (s,a,j,t)$
- $\Phi'(s) = \bigwedge_{(s_\mathsf{I},a,t_\mathsf{I}) \in T'} (a, t_\mathsf{I})$

That the constructed implementation is indeed an implementation of the given MTSD and that the running cost of the implementation starting from state s_I is the same as $v((s,\Phi(s)))$ is straightforward and can be proved as in the previous case.

Similarly, one can derive a strategy from an implementation so that its value does not worsen. We may safely assume that the implementation only implements durations as the extremal points of the controllable intervals. (Indeed, due to the original translation, Lemma 13, and the transformation back of Lemma 22, one can obtain an implementation with extremal points only that has the same or smaller cost.) The new transformation (strategy σ and mapping μ) is the same as the previous one of Lemma 22, the difference is that for history π ending in $(s, \varphi_1 \vee \varphi_2)$ we set $\sigma(\pi) = (s, \varphi_j)$ for an arbitrary $j \in \{1, 2\}$ such that $T(i_n) \models \varphi_j$ where $\mu(\pi') = i_0 \cdots i_n$ and π' is the longest prefix of π ending with the vertex $(s, \Phi(s))$. □