Modal Transition Systems with Weight Intervals

Line Juhl*, Kim G. Larsen, Jiří Srba[1]

Aalborg University, Department of Computer Science, Selma Lagerlöfs Vej 300, 9220 Aalborg Ø

Abstract

We propose *weighted modal transition systems*, an extension to the well-studied specification formalism of modal transition systems that allows to express both required and optional behaviours of their intended implementations. In our extension we decorate each transition with a weight interval that indicates the range of concrete weight values available to the potential implementations. In this way resource constraints can be modelled using the modal approach. We focus on two problems. First, we study the question of existence/finding the largest common refinement for a number of finite deterministic specifications and we show PSPACE-completeness of this problem. By constructing the most general common refinement, we allow for a stepwise and iterative construction of a common implementation. Second, we study a logical characterisation of the formalism and show that a formula in a natural weight extension of the logic CTL is satisfied by a given modal specification if and only if it is satisfied by all its refinements. The weight extension is general enough to express different sorts of properties that we want our weights to satisfy.

Keywords: modal transition systems, weighted transition systems, deterministic specifications, refinement, action-based CTL, model checking

1. Introduction

Modal transition systems [1] provide a specification formalism which can express both safety and liveness requirements of their implementations—labelled transition systems. This formalism allows for two kinds of transitions to be present, namely *required* (must) transitions and *allowed* (may) transitions. A rather loose specification can then be transformed into a concrete implementable system by a series of refinements. This idea of stepwise refinement is applicable for example for the development of embedded systems. Recent work on

*Corresponding author

Email addresses: linej@cs.aau.dk (Line Juhl), kgl@cs.aau.dk (Kim G. Larsen), srba@cs.aau.dk (Jiří Srba)

[1]Partially supported by Ministry of Education of the Czech Republic, project No. MSM 0021622419.

modal transition systems includes applications in several different areas like component-based software development [2, 3], interface theory [4, 5], modal abstractions and program analysis [6, 7, 8], and other areas [9, 10]. An overview article can be found in [11]. A similar concept has been studied also in the area of software product lines (see e.g. [12] and [13]), however, their notion of refinement is syntactic and different from the semantic refinement relation (based on the concepts of simulation/bisimulation) studied in the theory of modal transition systems.

We present an extension of modal transition systems called *weighted* modal transition systems that decorate each transition with an interval containing a range of weights. The idea of modelling quantitative aspects in transition systems is well studied. For example weighted transition systems (see e.g. the book [14]) are a known extension of standard labelled transition systems. Such systems are particularly useful for modelling resource constraints, which are often seen in embedded systems (e.g. fuel/power consumption, price). Weights therefore seem like a natural addition to modal transition systems, in order to combine the benefits of the 'modal' approach with the modelling of quantities. By allowing both negative and positive weights, we are furthermore able to model systems with both resource gains and losses.

Contrary to weighted transition systems, where transitions and/or states are labelled with specific weights, we decorate transitions with sets of weights. This adheres to the idea of a 'loose' specification, since a specification then determines the range of allowed weights instead of the precise weight. The refinement process will then rule out some of the weights, eventually ending up with an implementation containing the final concrete weight.

To motivate the use of weighted modal transition systems as a model for embedded systems, consider an ATM machine. Two clients might each give a specification (or requirements), detailing their allowed and required use of the machine, along with intervals specifying the acceptable power consumption for each option. This is demonstrated in Figure 1 (the two topmost systems). Here the dotted lines denote allowed behavior (i.e. the behaviour that a client is willing to perform), while the solid lines denote required behavior (i.e. the behaviour a client is insisting on). The interval attached to each transition is the interval where the power consumption (or some other cost) must lie in. As we can see, both clients require that a card is inserted. After the insertion, the clients only allow three actions, namely balance, withdraw and transfer. The balance option is required by the left client, while the right client requires that a withdrawal must be possible in an implementable system satisfying the specification. Even though the left client only specifies a withdrawal as optional, he/she requires that a PIN must be entered in order to continue. After the PIN is accepted, an amount can be withdrawn any number of times. The right client on the other hand specifies that a PIN is only optional, however, that each amount withdrawn must be preceded with re-entering the PIN.

An important problem is now to determine the existence of an implementation satisfying the needs of both clients and giving the exact power consumption for each option, fitting in the consumption requirements made by the clients.

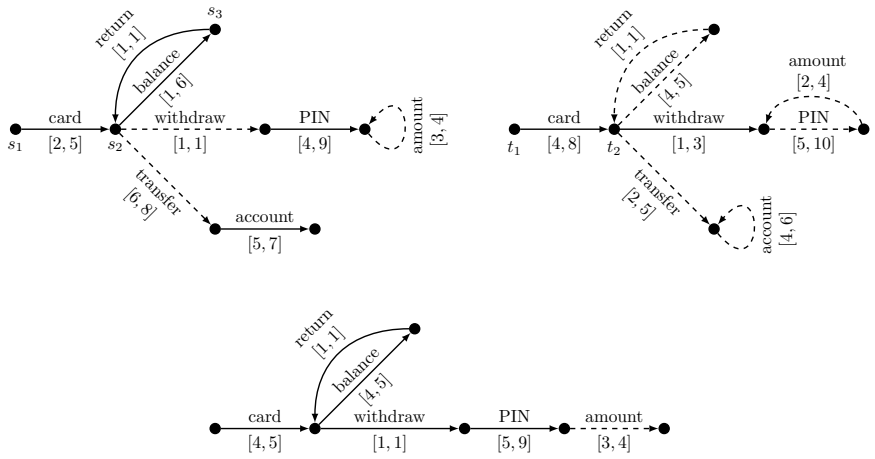

Figure 1: Two specifications of an ATM machine and their largest common refinement below.

We call such an implementation a *common implementation*. As it can be seen, the option of a transfer is allowed by both clients, but since their power consumption intervals are not overlapping, it is not possible to produce a specific system with a power consumption satisfying both clients. The transfer option is, however, not required by any of the clients, and can thus be ignored in a possible common implementation. Since insertion of the card, balance, withdrawal and PIN entering are required behavior for one or the other of the two specifications, these must be present in the implementable system. Instead of constructing just one common implementation, we aim at constructing the most permissive common refinement, so that this refinement encapsulates all common implementations. Figure 1 shows a most permissive common refinement below the two clients' specifications. After entering the PIN a withdrawal is only allowed once, since the right-hand side specification requires that a new amount specification is preceded by a PIN, while the left-hand side specification does not.

Considering the common refinement in Figure 1, one might be interested in knowing whether it is possible to withdraw some amount consuming between 10 and 20 energy units. Since a withdrawal consumes at least 13 energy units and at most 19 (adding the lower and upper interval bounds along the path), this is indeed possible. However, since the transition labelled 'amount' is only optional, some concrete implementations may leave it out. It is therefore desirable to develop a logical setting that guarantees that if some property is true for a given specification then it is also true for all its implementations.

Our contribution consists of a definition of weighted modal transition systems and an extension of the concepts related to modal transition systems to the weighted setting. This includes modal and thorough refinements and the definition of an implementation. Then we study the largest common refinement

problem of finite deterministic specifications. A construction computing the conjunction of a given number of finite deterministic specifications is presented, and we show that a given specification is their common refinement if and only if it refines the constructed largest common refinement. We further show that deciding whether a common refinement exists or not is a PSPACE-complete problem.

Our algorithm for the largest common refinement was inspired by the common implementation construction provided in [15]. However, we extend this technique to the weighted scenario and more importantly generalize the construction such that we construct the 'most permissive' common refinement, contrary to [15] where only the existence of a common implementation was studied. The maximality of our construction hence allows for a stepwise and iterative construction of a common implementation, which is desirable in many applications and was not possible with the previous algorithms.

We note that in this study we restrict ourselves to deterministic specifications as demonstrated, for example, in our running examples. There are two reasons that justify this choice. First of all, for nondeterministic specifications the two studied notions of thorough and modal refinement do not coincide and hence the refinement process, though sound, is not complete (see e.g. [11]). On the other hand for deterministic specifications, as advocated in the work by Henzinger and Sifakis [16, 17], modal refinement and modal composition are complete. More detailed analysis of this has been recently given in [15]. Second, in many practical cases, deterministic specifications are desirable and often used, and much of the recent work deals mainly with deterministic systems. For example in [16] the authors discuss two main challenges in embedded systems design: the challenge to build predictable systems, and that to build robust systems. They suggest how predictability can be formalized as a form of determinism, and robustness as a form of continuity.

Another problem we study in this article concerns finding a logical characterisation of weighted modal transition systems. By a natural extension of the action-based CTL we define, based on the work of De Nicola and Vaandrager [18], a weighted CTL logic for model checking weighted modal specifications. Compared to other weighted logics like [19] and [20], we allow to state arbitrary constraints on the prefixes of model executions and extend the semantics to deal with modal transition systems. On the other hand, we do not consider semiring interpretations of CTL formula quantifiers like in [19] and semiring semantics of MSO like in [20].

The definition of our logic is rather generic with respect to the choice for querying the weight constraints. Our main result shows that a specification satisfies a given formula of weighted CTL if and only if all its refinements satisfy the same formula, which is an important fact that justifies the choice of the logic and supports a step-wise model based development process. We discuss a few specializations of the generic logic to some concrete instances in order to argue for its applicability.

The article is organized as follows. Section 2 introduces the model of weighted modal transition systems, modal and thorough refinement relations and some

basic properties of the model. In Section 3 we study the problem of largest common refinement of a given set of finite deterministic specifications and among others prove PSPACE-completeness of the problem. In Section 4 we search for a logical characterisation of weigted modal transition systems. For this purpose we suggest a definition of a generic weighted CTL logic and argue for the soundness of this choice. Finally, Section 5 provides a short summary and mentions some of the open problems.

2. Definitions

We begin by extending the notion of modal transition systems (consult e.g. [1, 11]) by adding an interval to each transition in the specification. This set denotes the different values that the weight of the transition can be instantiated to in an implementation. We define $[n, m] = \{a \in \mathbb{Z} : n \leq a \leq m\}$ for $n \leq m$, $n, m \in \mathbb{Z} \cup \{-\infty, \infty\}$ to denote the closed interval between n and m, and use $\mathcal{I}$ to stand for the set of all such nonempty intervals.

Definition 1. A (interval) weighted modal transition system (WMTS) is a 5-tuple $M = (S, \Sigma, \dashrightarrow, \longrightarrow, \delta)$, where S is a set of states, Σ is an action alphabet, $\longrightarrow \subseteq \dashrightarrow \subseteq S \times \Sigma \times S$ and $\delta : (\dashrightarrow \cup \longrightarrow) \to \mathcal{I}$ assigns a weight interval to transitions. The relations $\dashrightarrow$ and $\longrightarrow$ are called the may and must transitions, respectively.

By the definition of δ we see that if (s, a, t) belongs to both $\dashrightarrow$ and $\longrightarrow$ then the weight intervals of the must and may transition are the same. This fact is important and implicitly used later on. It ensures the so-called *consistency*, meaning that any given modal specification is guaranteed to have an implementation.

We write $s \overset{a}{\dashrightarrow} t$ if $(s, a, t) \in \dashrightarrow$ and $s \overset{a,W}{\dashrightarrow} t$ if $e = (s, a, t) \in \dashrightarrow$ and $\delta(e) = W$, similarly for the elements of $\longrightarrow$. If no t exists such that $(s, a, t) \in \dashrightarrow$, we write $s \not\overset{a}{\dashrightarrow}$, similarly for must transitions. The class of all WMTSs is denoted by $\mathcal{W}$. An WMTS is *deterministic* if for all $s \in S$ and $a \in \Sigma$ there is at most one t such that $(s, a, t) \in \dashrightarrow$. The class of all deterministic WMTSs is denoted by $d\mathcal{W}$.

While a general WMTS models a specification giving a variety of weights and optional behaviour, an implementation (defined below) defines the precise behaviour of the system, including the precise weight of all transitions.

Definition 2 (Implementation). A WMTS is an implementation if $\longrightarrow = \dashrightarrow$ and all weight intervals are singletons. The class of all implementations is denoted by $i\mathcal{W}$.

To ease the notation, we often denote a WMTS $M = (S, \Sigma, \dashrightarrow, \longrightarrow, \delta)$ containing a state $s \in S$ as a pair, (s, M). Thus the notation $(s, M) \in \mathcal{W}$ is short hand notation for $M \in \mathcal{W}$ with s a state in M (the same applies to $i\mathcal{W}$ and $d\mathcal{W}$). The lowercase letters $s, t, \ldots$ are used for states (specifications)

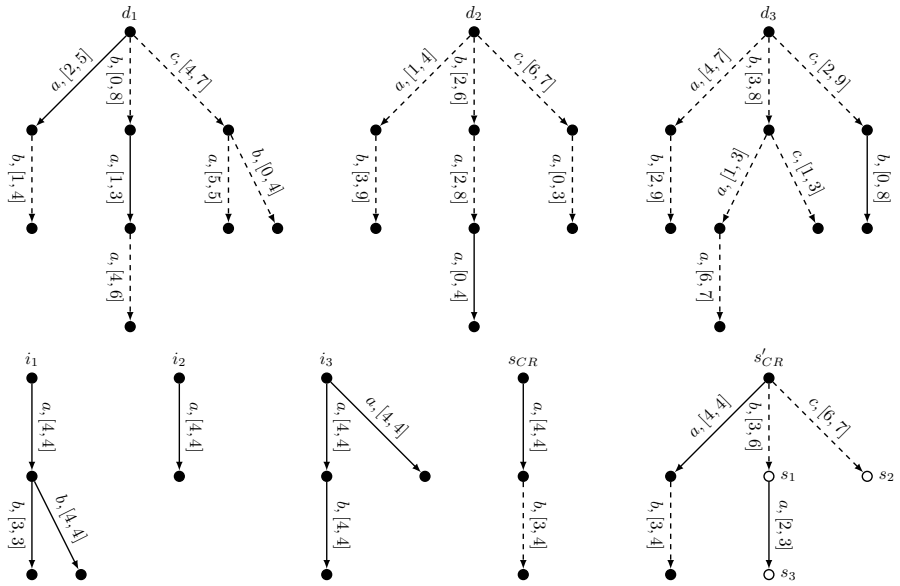

Figure 2: Different examples of weighted modal transition systems.

in general, while $i, j, \ldots$ are used for implementations and $d, e \ldots$ are used for deterministic specifications. Since every must transition is also a may transition, may transitions in figures will not be drawn between states if a must transition is already present.

Take a look at the examples in Figure 2 (ignore the systems s_{CR} and s'_{CR} for the moment). The three systems rooted with d_1, d_2 and d_3 are examples of weighted modal transition systems, all of them being deterministic. The systems rooted with i_1, i_2 and i_3 are examples of implementations where may and must transitions coincide and all intervals are singletons.

We can now define the refinement relation for WMTSs, a natural extension of the refinement relation on MTSs. Intuitively, a weight interval on a transition denotes the only acceptable weights allowed in an implementation. A refinement should therefore never allow any new weights to be added, eventually leading to an implementation with only singleton intervals. From now on, when using the term refinement we always refer to the modal refinement relation between two WMTSs as defined below.

Definition 3 (Modal refinement of WMTS). Let $(s_i, M_i) \in \mathcal{W}$ such that $M_i = (S_i, \Sigma, \dashrightarrow_i, \longrightarrow_i, \delta_i)$ for $1 \leq i \leq 2$. We say that s_1 modally refines s_2, written $(s_1, M_1) \leq_m (s_2, M_2)$ or simply $s_1 \leq_m s_2$ if M_1 and M_2 are clear from the context, if there is a refinement relation $R \subseteq S_1 \times S_2$ such that $(s_1, s_2) \in R$ and for each $(s, t) \in R$ and every $a \in \Sigma$:

1. whenever $s \overset{a,W}{\dashrightarrow}_1 s'$, then there exists $t \overset{a,V}{\dashrightarrow}_2 t'$ where $W \subseteq V$ such that $(s', t') \in R$, and

2. whenever $t \xrightarrow{a,V}_2 t'$, then there exists $s \xrightarrow{a,W}_1 s'$ where $W \subseteq V$ such that $(s', t') \in R$.

Hence (s, M) refines (t, N) ($s \leq_m t$) if it is possible to mimic must transitions in N by M, and it is possible to mimic may transitions in M by N. We say that a WMTS (s, M) is an *implementation of* a WMTS (t, N) if $(s, M) \in i\mathcal{W}$ and $(s, M) \leq_m (t, N)$. Notice that any WMTS has an implementation, for instance one can turn all may transitions into must transitions and pick an arbitrary weight from each interval as the singleton weight.

Consult again Figure 2. The systems i_1, i_2, i_3 and s_{CR} are all refinements of the specification d_1 (in fact also of d_2 and d_3). The first three refinements i_1, i_2 and i_3 are also implementations of d_1.

Remark 1. Notice that for two implementations $(i, I), (j, J) \in i\mathcal{W}$, the relation of modal refinement, $(i, I) \leq_m (j, J)$, corresponds to strong bisimulation (with the assumption that actions and weights are considered as observable pairs).

Definition 4 (Thorough refinement). Let (s, M) be a WMTS and define $[\![s]\!] = \{(i, I) \in i\mathcal{W} : (i, I) \leq_m (s, M)\}$, that is all possible refinements of (s, M) that are also implementations. For $(s, M), (t, N) \in \mathcal{W}$ we say that s thoroughly refines t, written $s \leq_t t$ (or $(s, M) \leq_t (t, N)$), if $[\![s]\!] \subseteq [\![t]\!]$.

The following lemma is easy to prove, but it is an important property that guarantees a sound stepwise refinement development methodology.

Lemma 1. *The relations $\leq_m$ and $\leq_t$ are both transitive.*

Proof. Let $(s, M), (u, O), (t, N) \in \mathcal{W}$. First the case of $\leq_m$. Assume two relations R_1 and R_2 according to Definition 3 showing that $s \leq_m u$ and $u \leq_m t$. It is easy to check that the relation R defined as

$$R = \{(s', t') : \exists u'.((s', u') \in R_1 \wedge (u', t') \in R_2)\}$$

is indeed a refinement relation according to Definition 3 and that $(s, t) \in R$.

We now consider $\leq_t$. Assume $s \leq_t u$ and $u \leq_t t$. This immediately implies that $[\![s]\!] \subseteq [\![u]\!] \subseteq [\![t]\!]$ and hence that $s \leq_t t$. □

We can now show that modal refinement implies thorough refinement.

Lemma 2. *For two WMTSs, (s, M) and (t, N), it holds that*

$$s \leq_m t \Rightarrow s \leq_t t.$$

Proof. Assume that $s \leq_m t$. If $i \leq_m s$ for an implementation i, then by Lemma 1 also $i \leq_m t$. Hence $[\![s]\!] \subseteq [\![t]\!]$ which means that $s \leq_t t$. □

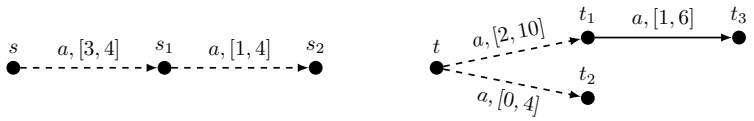

Figure 3: $s \leq_t t$, but $s \not\leq_m t$.

Notice that thorough refinement does not imply modal refinement. A counter-example can be seen in Figure 3. The figure is overtaken from [15] and intervals have been added, thus a counter-example already exists in the unweighted case. To show that $s \not\leq_m t$ we try to construct a relation R. For sure $(s,t) \in R$ must hold. Since $s \overset{a,[3,4]}{\dashrightarrow} s_1$ either (s_1,t_1) or (s_1,t_2) must belong to R. In the first case, $t_1 \overset{a,[1,6]}{\longrightarrow} t_3$ and therefore a must transition from s_1 must exist as well. Since this is not the case, we assume $(s_1,t_2) \in R$. Then the transition $s_1 \overset{a,[1,4]}{\dashrightarrow} s_2$ implies the existence of a may transition from t_2. This is also not the case, thus R cannot exist and $s \not\leq_m t$. On the other hand, every implementation of s can perform at most two consecutive a's with the weights either 3 or 4 for the first a-transition and 1, 2, 3 or 4 for the second a-transition. These implementations are also implementations of t, hence $s \leq_t t$.

However, if we restrict the refined specifications to be deterministic (or at least the right-hand side one) we get the following.

Lemma 3. *For $(s,M) \in \mathcal{W}$ and $(d,D) \in d\mathcal{W}$, it holds that*

$$s \leq_m d \Leftrightarrow s \leq_t d.$$

We omit the proof here, since it follows as a straightforward modification of the proof given in [15] by adding appropriate intervals to all transitions.

For the complexity results presented in the remainder of the paper we assume constant time interval operations (the encoding of integers is assumed binary).

In [21] it was shown that checking whether a finite modal transition system is thoroughly refined by another finite modal transition system is EXPTIME-complete. The thorough refinement problem for MTSs can be reduced to the same problem for WMTSs by adding the same singleton weight to all transitions. Hence the thorough refinement problem for finite WMTSs is also EXPTIME-hard. The algorithm presented in [21] for determining whether one MTS thoroughly refines another one can be easily extended to the weighted setting by adding appropriate checks for set inclusions of the weight intervals. This addition does not effect the running time of the algorithm, and the thorough refinement problem for finite WMTS is therefore decidable in EXPTIME as well.

On the contrary, the problem of deciding whether two finite weighted modal specifications are in the modal refinement relation is decidable in deterministic polynomial time using the standard greatest fixed-point computation, similarly as in the case of strong bisimulation (for efficient algorithms implementing this strategy see e.g. [22, 23]).

3. Largest Common Refinement

This section addresses the *largest common refinement problem* of finite deterministic specifications defined as follows: given a number of finite deterministic WMTSs, $(d_1, D_1), \ldots, (d_n, D_n)$, we want to find a specification $(s, M) \in \mathcal{W}$ such that $(s, M) \leq_m (d_j, D_j)$ for all j, $1 \leq j \leq n$, or to report that no such common refinement exists. Moreover, we are interested in constructing some largest common refinement refinement (s, M) such that any other common refinement of the given deterministic specifications refines (s, M). Notice that such a largest common refinement is not unique. In what follows, we implicitly assume that the given deterministic specifications $(d_1, D_1), \ldots, (d_n, D_n)$ are finite.

Figure 2 shows our running example. Our task is to construct the largest common refinement of the deterministic specifications d_1, d_2 and d_3.

Let $(d_1, D_1), \ldots, (d_n, D_n) \in d\mathcal{W}$ be n deterministic WMTSs. We will construct a specification $(s_{CR}, M_{CR}) \in \mathcal{W}$ and prove that (s_{CR}, M_{CR}) is the most general common refinement of $d_1, \ldots, d_n$. The state set of M_{CR} consists of n-tuples, $(e_1, \ldots, e_n)$, where every e_i belongs the the corresponding state set of D_i. Additionally some states in M_{CR} will be marked. Marked nodes represent situations where no common refinement exists. The pseudo-code in Alg. 1 constructs M_{CR}, a common refinement of the given specifications, or returns that no such refinement exists. The algorithm avoids the construction of the whole product space and returns only the reachable parts of such common refinement.

It is easy to see that the algorithm always terminates. The first repeat-loop (lines 3–22) runs until *Waiting* is empty, and in each iteration one element is removed from *Waiting*. Elements are also added to *Waiting*, however since D_i are finite for all i, and no removed element is added again to *Waiting*, this repeat-loop terminates. The second repeat-loop (lines 23–27) as well as the forall-loop (lines 30 and 31) also terminate due to the finiteness of the set S and finiteness of the $\longrightarrow$ relation.

Alg. 1 constructs M_{CR} by inspecting the n deterministic WMTSs and adding the needed states and transitions to M_{CR}, and furthermore marking states if these represent situations where no common refinement can exist. The marked set is expanded by adding all states, from which a path consisting of only must transitions leads to a marked state. If the state $(d_1, \ldots, d_n)$ is marked, no common refinement exists. If this is not the case, a common refinement exists and the most general common refinement is constructed by removing all marked states and transitions leading to marked states.

In our running example in Figure 2, given the input d_1, d_2 and d_3 Alg. 1 first constructs an intermediate specification s'_{CR} where the marked nodes s_1, s_2 and s_3 are drawn as circles. After removing them the algorithm returns the specification $(s_{CR}, M_{CR},)$.

Lemma 4. *If $(e_1, \ldots, e_n)$, a state in M_{CR}, has a path consisting only of must transitions leading to a state marked by Alg. 1 in the first repeat-loop, then $e_1, \ldots, e_n$ have no common refinement.*

Input: A finite number n of deterministic WMTSs, $(d_i, D_i) \in d\mathcal{W}$, where $D_i = (S_i, \Sigma, \dashrightarrow_i, \longrightarrow_i, \delta_i)$ for $i = 1, \ldots, n$.

Output: The string *"No common refinement exists"* or a $(s_{CR}, M_{CR}) \in d\mathcal{W}$, where $M_{CR} = (S, \Sigma, \dashrightarrow, \longrightarrow, \delta)$ s.t. $(s_{CR}, M_{CR}) \leq_m (d_i, D_i)$ for all i.

begin
  $S := \emptyset$; $\longrightarrow := \emptyset$; $\dashrightarrow := \emptyset$; $Marked := \emptyset$; $Waiting := \{(d_1, \ldots, d_n)\}$;
  repeat
    Select $(e_1, \ldots, e_n) \in Waiting$; $Waiting := Waiting \setminus \{(e_1, \ldots, e_n)\}$;
    $S := S \cup \{(e_1, \ldots, e_n)\}$;
    forall the $a \in \Sigma$ **do**
      if $\exists i : e_i \xrightarrow{a}_i f_i$ *and* $\forall j : e_j \overset{a}{\dashrightarrow}_j f_j$ **then**
        $temp := \delta_1((e_1, a, f_1)) \cap \ldots \cap \delta_n((e_n, a, f_n))$;
        if $temp = \emptyset$ **then**
          $Marked := Marked \cup \{(e_1, \ldots, e_n)\}$;
        else
          $\longrightarrow := \longrightarrow \cup \big\{\big((e_1, \ldots, e_n), a, (f_1, \ldots, f_n)\big)\big\}$;
          $\delta\big(\big((e_1, \ldots, e_n), a, (f_1, \ldots, f_n)\big)\big) := temp$;
          if $(f_1, \ldots, f_n) \notin S$ **then**
            $Waiting := Waiting \cup \{(f_1, \ldots, f_n)\}$;
      if $\forall i : e_i \overset{a}{\dashrightarrow}_i f_i$ **then**
        $temp := \delta_1((e_1, a, f_1)) \cap \ldots \cap \delta_n((e_n, a, f_n))$;
        if $temp \neq \emptyset$ **then**
          $\dashrightarrow := \dashrightarrow \cup \big\{\big((e_1, \ldots, e_n), a, (f_1, \ldots, f_n)\big)\big\}$;
          $\delta\big(\big((e_1, \ldots, e_n), a, (f_1, \ldots, f_n)\big)\big) := temp$;
          if $(f_1, \ldots, f_n) \notin S$ **then**
            $Waiting := Waiting \cup \{(f_1, \ldots, f_n)\}$;
      if $\exists i : e_i \xrightarrow{a}_i f_i$ *and* $\exists j : e_j \not\overset{a}{\dashrightarrow}_j$ **then**
        $Marked := Marked \cup \{(e_1, \ldots, e_n)\}$;
  until $Waiting = \emptyset$;
  repeat
    $Marked' := Marked$;
    forall the $(e, a, f) \in \longrightarrow$ **do**
      if $f \in Marked$ **then** $Marked := Marked \cup \{e\}$;
  until $Marked' = Marked$;
  if $(d_1, \ldots, d_n) \in Marked$ **then return** *"No common refinement exists"*
  $S := S \setminus Marked$;
  forall the $(e, a, f) \in \longrightarrow \cup \dashrightarrow$ *where* $f \in Marked$ **do**
    $\longrightarrow := \longrightarrow \setminus \{(e, a, f)\}$; $\dashrightarrow := \dashrightarrow \setminus \{(e, a, f)\}$;
  $s_{CR} := (d_1, \ldots, d_n)$; **return** (s_{CR}, M_{CR})

Algorithm 1: Construction of the most general common refinement.

PROOF. Assume that $(e_1, \ldots, e_n)$ is a state in M_{CR}, from which there is a path consisting of only must transitions leading to a state marked by Alg. 1 in the first repeat-loop. We show that the states $e_1, \ldots, e_n$ have no common refinement.

First observe that if $(e_1, \ldots, e_n) \xrightarrow{a,V} (f_1, \ldots, f_n)$ exists in M_{CR} we know that $e_i \xrightarrow{a,V_i} f_i$, where $V \subseteq V_i$ for at least one i and $e_j \overset{a,V_j}{\dashrightarrow} f_j$, where $V \subseteq V_j$ for all j (Alg. 1, line 6-13). Let p be any common refinement of $e_1, \ldots, e_n$. Assume therefore that n refinement relations R_j exist such that $(p, e_j) \in R_j$ for all j. Then p must have a $p \xrightarrow{a,W} q$ transition, where $W \subseteq V_j$ and $(q, f_j) \in R_j$ for all j. The fact that q is a refinement of f_i is clear by the second item in Definition 3, since $(p, e_i) \in R_i$ and $e_i \xrightarrow{a,V_i} f_i$, where $V \subseteq V_i$, forces a $p \xrightarrow{a,W} q$ transition, where $W \subseteq V_i$ and $(q, f_i) \in R_i$. The fact that q is also required to be a refinement of $f_1, \ldots, f_n$ is given by the first item in Definition 3, since $(p, e_j) \in R_j$ for all j and $p \overset{a,W}{\dashrightarrow} q$. Since $e_1, \ldots, e_n$ are deterministic, $e_j \overset{a,V_j}{\dashrightarrow} f_j$, where $W \subseteq V_j$ for all j are forced to be the matching transitions, thus requiring $(q, f_j) \in R_j$ to hold.

Next observe that if a state $(g_1, \ldots, g_n) \in \mathit{Marked}$ in Alg. 1 then either $(g_1, \ldots, g_n)$ has been marked at line 9 or line 21. If it is marked at line 9, then there exists some $a \in \Sigma$ and $f_1, \ldots, f_n$ such that $g_i \xrightarrow{a}_i f_i$ for at least one i and $\delta_1((g_1, a, f_1)) \cap \ldots \cap \delta_n((g_n, a, f_n)) = \emptyset$. Otherwise (if it was marked at line 21) there exists i such that $g_i \xrightarrow{a}_i f_i$ and there exists j such that $g_j \not\overset{a}{\dashrightarrow}_j$. Both cases imply that $g_1, \ldots, g_n$ have no common refinement. The first case is obvious, since the weight set of the matching transition in a common refinement must be contained in every $\delta_i((g_i, a, f_i))$ (due to determinism), but no transition with an empty weight set is allowed. The second case also leads to no common implementation, since $g_i \xrightarrow{a,V_i} f_i$ forces $p \xrightarrow{a,V} q$, where $V \subseteq V_i$ in the refinement, but $p \overset{a,V}{\dashrightarrow} q$ cannot be matched from g_j.

These two observations lead to our conclusion, since any common refinement p of $e_1, \ldots, e_n$, with refinement relations R_j and $(p, e_j) \in R_j$ for all j eventually fulfills $(q, g_j) \in R_j$ for some q, because of the path consisting of only must transitions. However, since $g_1, \ldots, g_n$ cannot have a common refinement, R_j cannot exist. □

By noting that Alg. 1 returns *"No common refinement exists"* only if there exists a path consisting of only must transitions from s_{CR} to a node marked before in the first repeat-loop and applying Lemma 4 to s_{CR} we have the following corollary.

Corollary 5. *If Alg. 1 returns "No common refinement exists" then the specifications $(d_1, D_1), \ldots, (d_n, D_n)$ have no common refinement.*

Lemma 6. *If $(d_1, D_1), \ldots, (d_n, D_n)$ have no common refinement then Alg. 1 returns "No common refinement exists".*

PROOF. We prove the contraposition. That is, assume that Alg. 1 does not return *"No common refinement exists"*. This implies that $(d_1, \ldots, d_n)$ is not a

marked state and that Alg. 1 returns (s_{CR}, M_{CR}). We want to construct relations $R_1, \ldots, R_n$ in order to show that s_{CR} is a common refinement of $d_1, \ldots, d_n$. We define the relations as

$$R_i = \{((e_1, \ldots, e_n), e_i) : (e_1, \ldots, e_n) \in M_{CR}\}$$

and continue to prove that these n relations fulfill the criteria of Definition 3. It is clear that $((d_1, \ldots, d_n), d_i) \in R_i$. Let $((e_1, \ldots, e_n), e_i) \in R_i$ and consider a must transition $e_i \xrightarrow{a,V_i} f_i$. Since by assumption the algorithm returns (s_{CR}, M_{CR}) and $(e_1, \ldots, e_n) \in M_{CR}$, $(e_1, \ldots, e_n)$ is not marked. Therefore $e_j \overset{a,V_j}{\dashrightarrow} f_j$ exists for all j and the transition $(e_1, \ldots, e_n) \xrightarrow{a,\bigcap_j V_j} (f_1, \ldots, f_n)$ is added to M_{CR} in Alg. 1, line 11-13. Furthermore, $(f_1, \ldots, f_n)$ is not marked (and thus not removed), since otherwise $(e_1, \ldots, e_n)$ would have been marked during the repeat loop in line 23-27, a contradiction. Hence $(f_1, \ldots, f_n)$ is a state in M_{CR} and $((f_1, \ldots, f_n), f_i) \in R_i$ as required.

On the other hand, consider a may transition $(e_1, \ldots, e_n) \overset{a,V}{\dashrightarrow} (f_1, \ldots, f_n)$. By construction $e_i \overset{a,V_i}{\dashrightarrow} f_i$, where $V \subseteq V_i$ for all i. Hence $((f_1, \ldots, f_n), f_i) \in R_i$ for all i as required. □

With the above lemmas and Alg. 1 we have the following complexity result.

Theorem 7. *The problem of existence of a common refinement for a given number of finite deterministic specifications is PSPACE-complete.*

Proof. In order to show containment in PSPACE, Cor. 5 and Lemma 6 give us that the existence of a common refinement is equivalent to the question whether the state s_{CR} gets marked by Alg. 1. In other words, this is equivalent to the question whether there is a must-path from s_{CR} to some state marked directly in line 9 or 21 of the algorithm. Such a path can be nondeterministically guessed on the fly (without constructing the whole state-space) and by Savitch's theorem this implies the containment in PSPACE.

The hardness result follows directly from PSPACE-hardness of the common implementation problem for unweighted MTSs shown in [15]. □

The last theorem states that M_{CR} is the largest common refinement.

Theorem 8 (Maximality of M_{CR}). *If Alg. 1 returns (s_{CR}, M_{CR}) then for every $(t, N) \in \mathcal{W}$ such that $(t, N) \leq_m (d_i, D_i)$ for all i, $1 \leq i \leq n$, it holds that $(t, N) \leq_m (s_{CR}, M_{CR})$.*

Proof. Let $(t, N) \in \mathcal{W}$ be a common refinement of $(d_1, D_1), \ldots, (d_n, D_n)$. This means that there exist n relations $R_1, \ldots, R_n$ satisfying the conditions in Definition 3. We construct a new relation Q satisfying the same conditions in order to prove that $(t, N) \leq_m (s_{CR}, M_{CR})$. The relation Q is defined as follows:

$$(s, (e_1, \ldots, e_n)) \in Q \text{ if and only if } (s, e_i) \in R_i \text{ for all } i,\ 1 \leq i \leq n.$$

We observe that $(t,(d_1,\ldots,d_n)) \in Q$, since $(t,d_i) \in R_i$ for all i. This satisfies the first condition in the refinement definition. Let now $(s,(e_1,\ldots,e_n)) \in Q$ and consider what happens in case of must and may transitions.

Consider a transition $s \overset{a,V}{\dashrightarrow} s'$. Then for all i we have that for all e_i such that $(s,e_i) \in R_i$ there exists f_i such that $e_i \overset{a,W_i}{\dashrightarrow} f_i$ with $V \subseteq W_i$ and $(s',f_i) \in R_i$. This implies that $q = (e_1,\ldots,e_n) \overset{a,W_1\cap\ldots\cap W_n}{\dashrightarrow} (f_1,\ldots,f_n)$ is a transition in M_{CR}. Since $V \subseteq W_1 \cap \ldots \cap W_n$ and $(s',f_i) \in R_i$ for all i, then $(s',(f_1,\ldots,f_n)) \in Q$ as desired. Notice that $(f_1,\ldots,f_n)$ cannot be a node which was removed in Alg. 1 since this would imply that $(f_1,\ldots,f_n)$ is marked. The states $f_1,\ldots,f_n$ would not therefore have a common refinement by Lemma 4. This contradicts the fact that $(s',f_i) \in R_i$ for all i.

Consider now a transition $(e_1,\ldots,e_n) \xrightarrow{a,V} (f_1,\ldots,f_n)$. By construction of M_{CR} we know that there exists at least one e_j such that $e_j \xrightarrow{a,W_j} f_j$ and that $e_i \overset{a,W_i}{\dashrightarrow} f_i$ exist for all i (Alg. 1, line 6-13). Since $(d_1,D_1),\ldots,(d_n,D_n)$ are deterministic, f_i (for all i) are unique. Now because $e_j \xrightarrow{a,W_j} f_j$ and $(s,e_j) \in R_j$, we get that $s \xrightarrow{a,W} s'$ with $W \subseteq W_j$ such that $(s',f_j) \in R_j$. This, however, also means that $s \overset{a,W}{\dashrightarrow} s'$ and because $(s,e_i) \in R_i$ for all i, we get that $(s',f_i) \in R_i$ and $W \subseteq W_i$ for all i. Notice that $V = W_1 \cap \ldots \cap W_n$, so $W \subseteq V$ holds. Thus $(s',(f_1,\ldots,f_n)) \in Q$ by the definition of Q. □

As a corollary and due to the transitivity of $\leq_m$ (Lemma 1), Cor. 5, Lemma 6 and the maximality of M_{CR} (Thm. 8) we get the main result.

Corollary 9. *Let $(d_1,D_1),\ldots,(d_n,D_n)$ be finite deterministic WMTSs and assume that Alg. 1 returns a specification (s_{CR},M_{CR}). A specification $(s,M) \in \mathcal{W}$ is a common refinement of the specifications $(d_1,D_1),\ldots,(d_n,D_n)$ if and only if $(s,M) \leq_m (s_{CR},M_{CR})$.*

This theorem allows us to find a common refinement (and thus also a common implementation) in a stepwise and iterative manner. Consider Figure 4. Here (s_{CR},M_{CR}) is constructed by giving $(d_1,D_1),\ldots,(d_n,D_n) \in d\mathcal{W}$ as input to Alg. 1. The specification (s^n_{CR},M^n_{CR}) is, on the other hand, constructed by using Alg. 1 iteratively. First $(d_1,D_1),(d_2,D_2)$ is given as input and then the output, (s^2_{CR},M^2_{CR}), and (d_3,D_3) is used as input, continuing in this way until the last received output and (d_n,D_n) is given as input, finally outputting (s^n_{CR},M^n_{CR}). The theorem below states that both applications of the algorithm lead to the same set of possible implementations.

Theorem 10. *Let $(d_1,D_1),\ldots,(d_n,D_n)$ be finite deterministic WMTSs and assume that (s_{CR},M_{CR}) and (s^n_{CR},M^n_{CR}) are the specifications obtained as illustrated in Figure 4. Then $[\![s_{CR}]\!] = [\![s^n_{CR}]\!]$.*

Proof. Using Corollary 9 gives us that $(s^n_{CR},M^n_{CR}) \leq_m (s_{CR},M_{CR})$. The other direction is an easy induction in n, the number of deterministic specifications. As a base case we have $n = 2$. The two uses of Alg. 1 are here

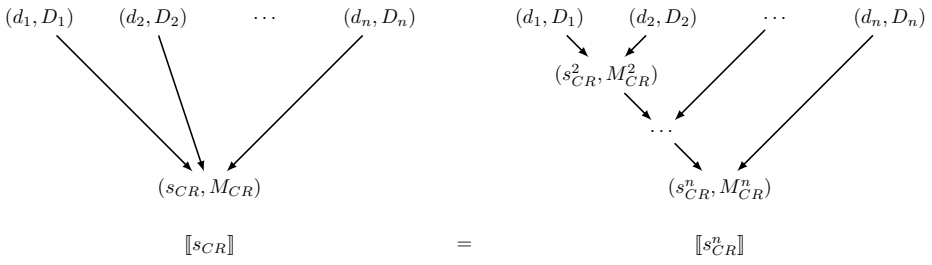

Figure 4: Two ways of using Alg. 1 yielding the same result.

equal, and the theorem follows. For the induction step assume $(s_{CR}, M_{CR}) \leq_m (s^{n-1}_{CR}, M^{n-1}_{CR})$ holds. Notice that now $(s_{CR}, M_{CR}) \leq_m (s^n_{CR}, M^n_{CR})$ also holds, since (s^n_{CR}, M^n_{CR}) is the output when Alg. 1 is given $(s^{n-1}_{CR}, M^{n-1}_{CR})$ and (d_n, D_n) as input and Corollary 9 is thus applicable. Lemma 2 now implies $[\![s_{CR}]\!] = [\![s^n_{CR}]\!]$. □

This result is more general than the algorithm in [15], which checks only for the existence of a common implementation of (unweighted) modal specifications. The algorithm presented here furthermore constructs the most permissive common refinement and provides support for step-wise development of systems.

4. Logical Characterisation

In the previous section we discussed algorithms for constructing some largest common refinement for a given set of weighted deterministic modal specifications. Now we shall turn our attention to a logical characterisation of weighted modal transition systems. We define an extension of the well-known CTL logic [24] that will allow us to state logical queries that include constraints about the weights along the finite and infinite paths. There are several well justified choices for the definition of the constraints on the paths. We provide a meta-definition of a general constraint form which specializes to many useful constraint choices. Our main result is that as long as a certain monotonicity property is preserved, a specification satisfies a given logical formula if and only if all its refinements do. This result can be understood as a soundness principle for the suggested logic.

We start with the definition of must-/may-paths in weighted modal transition systems.

Definition 5 (Path). A *must-path* in a WMTS $M = (S, \Sigma, \dashrightarrow, \longrightarrow, \delta)$ is a finite or infinite sequence π of transitions of the form

$$\pi = s_1 \xrightarrow{a_1} s_2 \xrightarrow{a_2} s_3 \xrightarrow{a_3} \dots \ .$$

A must-path is *maximal* if it is infinite or it ends in a state with no outgoing may transitions (and hence of course also no outgoing must transitions). The set of all maximal must-paths starting from a state s is denoted by $\mathsf{maxmustP}(s)$.

Similarly, a *may-path* is a finite or infinite sequence π of transitions of the form

$$\pi = s_1 \xrightarrow{a_1} s_2 \xrightarrow{a_2} s_3 \xrightarrow{a_3} \dots \ .$$

A may-path is *maximal* if it is infinite or it ends in a state with no outgoing must transition (note that outgoing may transitions are allowed). The set of all maximal may-paths starting from a state s is denoted by $\mathsf{maxmayP}(s)$. Notice that a must-path is not necessarily a prefix of a maximal must-path and that a maximal may-path may be a strict prefix of another maximal may-path.

Given a must- or may-path in the form above, the notation $\pi[j]$ denotes the j'th state of the path, that is $\pi[j] = s_j$.

For specifying logical properties we suggest a notion of weighted action-based CTL (WCTL), a particular extension of CTL (see e.g. [25]). The action-based syntax is based on the work of De Nicola and Vaandrager [18] which introduces an action-labelled next operator. In [18] they discuss a close relationship between action-based and state-based logics (see also [26]). We further extend their logic such that it can be interpreted over modal transition systems and we add a generic weight constraint function in order to reason about the cost of the transitions and demonstrate a few examples of well-justified weight constraint functions.

Let us first define the so-called *action formulae*:

$$\chi, \chi' ::= \mathsf{true} \mid \ a \ \mid \neg\chi \mid \chi \wedge \chi'$$

where $a \in \Sigma$ ranges over the actions of a given WMTS. The semantics to action formulae is given by the following satisfaction relation ($a, b \in \Sigma$):

$$\begin{array}{lll} a \models \mathsf{true} & & \\ a \models b & \text{iff} & a = b \\ a \models \neg\chi & \text{iff} & a \not\models \chi \\ a \models \chi \wedge \chi' & \text{iff} & a \models \chi \text{ and } a \models \chi' \ . \end{array}$$

The *(state) formulae* of WCTL are now generated by the following abstract syntax:

$$\begin{aligned} \varphi, \varphi_1, \varphi_2 ::= \ & \mathsf{true} \mid \mathsf{false} \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \vee \varphi_2 \mid \mathsf{EX}^{\mathsf{c}}_{\chi}\, \varphi \mid \mathsf{AX}^{\mathsf{c}}_{\chi}\, \varphi \\ & \mid \mathsf{E}\,(\varphi_1\, \mathsf{U}^{\mathsf{c}}_{\chi}\, \varphi_2) \mid \mathsf{A}\,(\varphi_1\, \mathsf{U}^{\mathsf{c}}_{\chi}\, \varphi_2) \mid \mathsf{E}\,(\varphi_1\, \mathsf{R}^{\mathsf{c}}_{\chi}\, \varphi_2) \mid \mathsf{A}\,(\varphi_1\, \mathsf{R}^{\mathsf{c}}_{\chi}\, \varphi_2) \end{aligned}$$

where χ ranges over action formulae and $\mathsf{c} : \mathcal{I}^* \cup \mathcal{I}^\omega \to \{0, 1\}$ is a constraint function assigning 0 (false) or 1 (true) to any finite and infinite sequence of weight intervals (for the definition of $\mathcal{I}$ see the first paragraph of Section 2). We moreover require that c satisfies the following monotonicity property:

$$\text{if } \mathsf{c}(w_1, w_2, \dots) = 1 \text{ then } \mathsf{c}(w_1', w_2', \dots) = 1 \text{ for any } w_i' \subseteq w_i \text{ for all } i.$$

This means that if some sequence of intervals is acceptable by the constraint function, so will be the sequence containing any subintervals. By $\mathcal{L}$ we denote the set of all WCTL formulae.

This syntax is similar to the standard action-based CTL. The main difference is the superscript 'c' attached to the next, until and release operators. For example, $\mathsf{E}\ (\varphi_1\ \mathsf{U}^{\mathsf{c}}_{\chi}\ \varphi_2)$ holds in a state s if there exists a maximal must-path which satisfies that φ_2 holds in some state along the path, φ_1 holds in all states prior to that state, the actions on the subpath where φ_1 holds satisfy χ and c is true for the sequence of intervals belonging to the subpath where φ_1 holds. The reason for choosing a must-path is that the existence of such a path in the specification will guarantee its existence also in any of its refinements. Similarly, the formula $\mathsf{E}\ (\varphi_1\ \mathsf{R}^{\mathsf{c}}_{\chi}\ \varphi_2)$ holds in a state s if there exists a maximal must-path which satisfies that φ_2 holds in all states along the path, a requirement that is dropped as soon as φ_1 holds, the actions on the path where φ_2 holds satisfy χ and c is true for the sequence of intervals belonging to the path where φ_2 holds (hence the need for the constraint function to be defined over infinite sequences of intervals too). For the path quantifier A, the temporal operators U and R have a similar meaning, only in this case we require that all maximal may-paths satisfy these properties.

The semantics of WCTL formulae is then interpreted over the states of a WMTS. Let $M = (S, \Sigma, \dashrightarrow, \longrightarrow, \delta) \in \mathcal{W}$, χ range over action formulae, $s \in S$, and φ, φ_1 and φ_2 be formulae from $\mathcal{L}$. The satisfaction relation $\models$ is defined by

$$
\begin{array}{lll}
s \models \mathsf{true} & & \\
s \not\models \mathsf{false} & & \\
s \models \varphi_1 \wedge \varphi_2 & \text{iff} & s \models \varphi_1 \text{ and } s \models \varphi_2 \\
s \models \varphi_1 \vee \varphi_2 & \text{iff} & s \models \varphi_1 \text{ or } s \models \varphi_2 \\
s \models \mathsf{EX}^{\mathsf{c}}_{\chi}\ \varphi & \text{iff} & \exists (s \xrightarrow{a} s') : a \models \chi\ \wedge\ s' \models \varphi\ \wedge\ \mathsf{c}(\delta(s,a,s')) = 1 \\
s \models \mathsf{AX}^{\mathsf{c}}_{\chi}\ \varphi & \text{iff} & \forall (s \overset{a}{\dashrightarrow} s') \text{ where } a \models \chi : s' \models \varphi\ \wedge\ \mathsf{c}(\delta(s,a,s')) = 1 \\
s \models \mathsf{E}\ (\varphi_1\ \mathsf{U}^{\mathsf{c}}_{\chi}\ \varphi_2) & \text{iff} & \exists \pi = s_1 \xrightarrow{a_1} s_2 \xrightarrow{a_2} \ldots \in \mathsf{maxmustP}(s) : \\
 & & \exists i \geq 1 : s_i \models \varphi_2 \\
 & & \wedge\ \forall j \in \{1, \ldots, i-1\} : (s_j \models \varphi_1\ \wedge\ a_j \models \chi) \\
 & & \wedge\ \mathsf{c}(\delta(s_1,a_1,s_2), \delta(s_2,a_2,s_3), \ldots, \delta(s_{i-1},a_{i-1},s_i)) = 1 \\
s \models \mathsf{A}\ (\varphi_1\ \mathsf{U}^{\mathsf{c}}_{\chi}\ \varphi_2) & \text{iff} & \forall \pi = s_1 \overset{a_1}{\dashrightarrow} s_2 \overset{a_2}{\dashrightarrow} \ldots \in \mathsf{maxmayP}(s) \\
 & & \text{where } a_i \models \chi \text{ for all } i : \\
 & & \exists i \geq 1 : s_i \models \varphi_2\ \wedge\ (\forall j \in \{1, \ldots, i-1\} : s_j \models \varphi_1) \\
 & & \wedge\ \mathsf{c}(\delta(s_1,a_1,s_2), \delta(s_2,a_2,s_3), \ldots, \delta(s_{i-1},a_{i-1},s_i)) = 1
\end{array}
$$

$$
\begin{aligned}
s \models \mathsf{E}\ (\varphi_1\ \mathsf{R}^{\mathsf{c}}_{\chi}\ \varphi_2) \quad \text{iff} \quad & \exists \pi = s_1 \xrightarrow{a_1} s_2 \xrightarrow{a_2} \ldots \in \mathsf{maxmustP}(s) : \\
& \Big(\forall k \geq 1 : s_k \models \varphi_2 \ \wedge\ a_k \models \chi \\
& \wedge\ \mathsf{c}(\delta(s_1, a_1, s_2), \delta(s_2, a_2, s_3), \ldots) = 1\Big) \\
& \vee \\
& \Big(\exists i \geq 1 : s_i \models \varphi_1 \wedge \forall j \in \{1, \ldots, i\} : (s_j \models \varphi_2 \wedge a_j \models \chi) \\
& \wedge\, \mathsf{c}(\delta(s_1, a_1, s_2), \delta(s_2, a_2, s_3), \ldots, \delta(s_{i-1}, a_{i-1}, s_i)) = 1\Big) \\
s \models \mathsf{A}\ (\varphi_1\ \mathsf{R}^{\mathsf{c}}_{\chi}\ \varphi_2) \quad \text{iff} \quad & \forall \pi = s_1 \overset{a_1}{\dashrightarrow} s_2 \overset{a_2}{\dashrightarrow} \ldots \in \mathsf{maxmayP}(s) \\
& \text{where } a_i \models \chi \text{ for all } i : \\
& \Big(\forall k \geq 1 : s_k \models \varphi_2 \\
& \wedge\ \mathsf{c}(\delta(s_1, a_1, s_2), \delta(s_2, a_2, s_3), \ldots) = 1\Big) \\
& \vee \\
& \Big(\exists i \geq 1 : s_i \models \varphi_1\ \wedge\ (\forall j \in \{1, \ldots, i\} : s_j \models \varphi_2) \\
& \wedge\, \mathsf{c}(\delta(s_1, a_1, s_2), \delta(s_2, a_2, s_3), \ldots, \delta(s_{i-1}, a_{i-1}, s_i)) = 1\Big).
\end{aligned}
$$

If the system M is not clear from the context, we also use the notation $(s, M) \models \varphi$ meaning that $s \models \varphi$ where s a state in M. We remark that for the cases of $\mathsf{E}\ (\varphi_1\ \mathsf{U}^{\mathsf{c}}_{\chi}\ \varphi_2)$ and $\mathsf{A}\ (\varphi_1\ \mathsf{R}^{\mathsf{c}}_{\chi}\ \varphi_2)$ we may consider also paths that are not necessarily maximal, but for the sake of techincal conveniences we restrict ourself to maximal runs in all cases.

Notice that, as usual, we can express the temporal modalities 'eventually' ($\mathsf{F}^{\mathsf{c}}_{\chi}$) and 'always' ($\mathsf{G}^{\mathsf{c}}_{\chi}$).

$$
\begin{aligned}
\mathsf{EF}^{\mathsf{c}}_{\chi}\ \varphi &\equiv \mathsf{E}\ (\mathsf{true}\ \mathsf{U}^{\mathsf{c}}_{\chi}\ \varphi) \\
\mathsf{AF}^{\mathsf{c}}_{\chi}\ \varphi &\equiv \mathsf{A}\ (\mathsf{true}\ \mathsf{U}^{\mathsf{c}}_{\chi}\ \varphi) \\
\mathsf{EG}^{\mathsf{c}}_{\chi}\ \varphi &\equiv \mathsf{E}\ (\mathsf{false}\ \mathsf{R}^{\mathsf{c}}_{\chi}\ \varphi) \\
\mathsf{AG}^{\mathsf{c}}_{\chi}\ \varphi &\equiv \mathsf{A}\ (\mathsf{false}\ \mathsf{R}^{\mathsf{c}}_{\chi}\ \varphi)
\end{aligned}
$$

The generic definition of the constraint function c can be specialized in order to express a variety of interesting properties that we want our sequences of intervals to fulfil. We will now give two examples.

Example 1. Consider the ATM machines shown in Figure 1 in Section 1. It might be worth knowing, for example, whether or not in any implementation it is possible to check the balance without entering PIN while consuming between 0 and 15 units of power. We thus want to reason about the accumulated energy along a given path. For this purpose, let the constraint function c_S be given as

$$
\mathsf{c}_S(w_1, w_2, \ldots) = \begin{cases} 1 & \text{if } \left[\sum_{i=1} n_i, \sum_{i=1} m_i\right] \subseteq S \\ 0 & \text{otherwise,} \end{cases}
$$

where $w_i = [n_i, m_i]$ and S is a given set of elements from $\mathbb{Z} \cup \{-\infty, \infty\}$. The constraint function then returns 1 if the interval containing all possible sums of weights belonging to each interval is contained in the set S, and 0 otherwise.

Notice that if $\mathsf{c}(w_1, w_2, \ldots) = 1$ then $\mathsf{c}(w_1', w_2', \ldots) = 1$ for any $w_i' \subseteq w_i$ for all i, since smaller intervals do not give rise to any new accumulated sums, thus still preserving the requirement of being a subset of S.

Using this constraint function one can specify that every possible total accumulated weight along some path should be contained in $[0, 15]$, as required in the ATM machine. The WCTL formula to check this is

$$\varphi \equiv \mathsf{E}\left(\mathsf{true}\ \mathsf{U}^{\mathsf{c}_{[0,15]}}_{\neg \mathrm{PIN}}\ \left(\ \mathsf{EX}^{\mathsf{c}_{[-\infty,\infty]}}_{\mathrm{return}}\ \mathsf{true}\right)\right) \equiv \mathsf{EF}^{\mathsf{c}_{[0,15]}}_{\neg \mathrm{PIN}}\ \left(\ \mathsf{EX}^{\mathsf{c}_{[-\infty,\infty]}}_{\mathrm{return}}\ \mathsf{true}\right),$$

stating that there exists a path where the action 'return' is enabled in some state in the future (with an arbitrary cost), and that we until reaching the state enabling 'return' must not take a transition with action 'PIN' and that the accumulated cost on this subpath (where the balance is checked) must consume between 0 and 15 power units.

Consulting the leftmost specification in Figure 1 we see that $s_1 \models \varphi$ holds, since a must-path $s_1 \overset{\mathrm{card},[2,5]}{\longrightarrow} s_2 \overset{\mathrm{balance},[1,6]}{\longrightarrow} s_3$ exists, and from s_3 an outgoing must transition with the action 'return' is required. Any implementation would therefore also require these three transitions.

On the other hand, $t_1 \not\models \varphi$, since in the specification the 'balance' transition is only optional, and thus might not be present in an implementation.

Example 2. As another example consider a specification of a gas tank, able to both gain and lose gas. In this case we are interested in keeping the volume of gas in the tank between some interval at all times, since the tank would otherwise explode or have a too low volume. It is therefore not adequate to only consider the volume at the end of a given path. All subpaths must also fulfil the interval bound (note that we allow for negative weights). We therefore define c_S as

$$\mathsf{c}_S(w_1, w_2, \ldots) = \begin{cases} 1 & \text{if } \forall j \geq 1 : \left[\sum_{i=1}^{j} n_i, \sum_{i=i}^{j} m_i\right] \subseteq S \\ 0 & \text{otherwise,} \end{cases}$$

where $w_i = [n_i, m_i]$ and S is a given set of elements from $\mathbb{Z} \cup \{-\infty, \infty\}$. In this way we specify, using the operator G, that the volume of the tank must be between 0 and 100 everywhere along all potentially infinite paths where the action emergency is not taken by

$$\mathsf{AG}^{\mathsf{c}_{[0,100]}}_{\neg \mathrm{emergency}}\ \mathsf{true}\ .$$

If emergency is triggered along some path, the tank will shut down and the volume need not be guarded any more.

This kind of weight constraint function proved useful e.g. in [27] where the existence of such an infinite path was studied in the context of weighted timed automata. Observe again that if $\mathsf{c}(w_1, w_2, \ldots) = 1$ then $\mathsf{c}(w_1', w_2', \ldots) = 1$ for any $w_i' \subseteq w_i$ for all i.

We shall now formulate a technical lemma that will be used in the proof of the main theorem of this section.

Lemma 11. *Let* $(s_1, M), (t_1, N) \in \mathcal{W}$ *and* $(s_1, M) \leq_m (t_1, N)$.

1. *If* $\pi_N \in \mathsf{maxmustP}(t_1)$ *then there exists* $\pi_M \in \mathsf{maxmustP}(s_1)$ *of the same length as* π_N*, such that* $(\pi_M[i], M) \leq_m (\pi_N[i], N)$ *for all* i.
2. *If* $\pi_M \in \mathsf{maxmayP}(s_1)$ *then there exists* $\pi_N \in \mathsf{maxmayP}(t_1)$ *of the same lento as* π_M*, such that* $(\pi_M[i], M) \leq_m (\pi_N[i], N)$ *for all* i.

Moreover, in both cases the weight intervals on the path π_M *are subintervals of the corresponding intervals on the path* π_N.

Proof. Let $(s_1, M), (t_1, N) \in \mathcal{W}$ and assume that $s_1 \leq_m t_1$.

1. Let $\pi_N = t_1 \xrightarrow{a_1,V_1} t_2 \xrightarrow{a_2,V_2} t_3 \xrightarrow{a_3,V_3} \dots$ be a maximal must-path in N. We want to show that there exists a maximal must-path $\pi_M = s_1 \xrightarrow{a_1,W_1} s_2 \xrightarrow{a_2,W_2} s_3 \xrightarrow{a_3,W_3} \dots$ in M such that $s_i \leq_m t_i$ and $W_i \subseteq V_i$ for all i. We prove this by induction. By induction hypothesis we assume that there is a path $s_1 \xrightarrow{a_1,W_1} s_2 \xrightarrow{a_2,W_2} \dots \xrightarrow{a_{j-1},W_{j-1}} s_j$ in M for $j > 1$ such that $s_i \leq_m t_i$ and $W_i \subseteq V_i$ for all i, $1 \leq i \leq j$. (The base case where $j = 1$ requires that $s_1 \leq_m t_1$ which is true by the assumption of the lemma.) We now distinguish two cases (recall that π_N is a maximal must-path).

   - *Case where* t_j *has no outgoing may transitions.* Because $s_j \leq_m t_j$ we get that s_j cannot have any outgoing transitions either and a maximal must-path in N was matched by a maximal must-path in M as required.

   - *Case where* $t_j \xrightarrow{a_j,V_j} t_{j+1}$. Because $s_j \leq_m t_j$ there must be a transition $s_j \xrightarrow{a_j,W_j} s_{j+1}$ such that $s_{j+1} \leq_m t_{j+1}$ and $W_j \subseteq V_j$. Hence there is a path $s_1 \xrightarrow{a_1,W_1} s_2 \xrightarrow{a_2,W_2} \dots \xrightarrow{a_j,W_j} s_{j+1}$ in M such that $s_i \leq_m t_i$ and $W_i \subseteq V_i$ for all i, $1 \leq i \leq j+1$, as required by the induction.

2. Let $\pi_M \in \mathsf{maxmayP}(s_1)$ be a maximal may-path in M. We want to find a matching maximal may-path in N. The arguments are symmetric as in the proof of part 1. Notice that the maximality of π_M in case of a finite path, i.e. the absence of any must transition at the end of the path, implies that the path π_N constructed in a similar manner as in part 1. is also maximal (the presence of a must transition at its last state would enforce the presence of a must transition in the last state of π_M). □

Given a specification $(s, M) \in \mathcal{W}$ and a WCTL formula φ such that $(s, M) \models \varphi$, the following theorem shows that any refinement also satisfies φ. This problem is closely related to generalized model checking (as defined in [28]), which asks, given $(s, M) \in \mathcal{W}$ and $\varphi \in \mathcal{L}$, does there exists an implementation $(i, I) \in [\![s]\!]$, such that $(i, I) \models \varphi$. In our case we, on the other hand, consider the validity problem for all refinements—does any refinement fulfil φ?

Theorem 12. *Let $(t, N) \in \mathcal{W}$ and let $\varphi \in \mathcal{L}$ be a WCTL formula. Then $(t, N) \models \varphi$ if and only if $(s, M) \models \varphi$ for all (s, M) s.t. $(s, M) \leq_m (t, N)$.*

PROOF. The 'if' part is trivial since (t, N) is a refinement of itself. We prove 'only if' below.

Let $(s, M), (t, N) \in \mathcal{W}$ be two weighted modal transition systems, where $M = (S_M, \Sigma, \dashrightarrow, \longrightarrow, \delta_M)$ and $N = (S_N, \Sigma, \dashrightarrow, \longrightarrow, \delta_N)$. We show that for any formula $\varphi \in \mathcal{L}$:

$$\text{if } (s, M) \leq_m (t, N) \text{ and } (t, N) \models \varphi \text{ then } (s, M) \models \varphi \ . \tag{1}$$

The proof is by structural induction over the structure of the formula φ.
Induction basis: The cases $\varphi = \mathsf{true}$ and $\varphi = \mathsf{false}$ are trivial.
Induction step: Assume φ_1 and φ_2 are state formulae for which (1) hold.

- $\varphi = \varphi_1 \wedge \varphi_2$:

  Since the induction hypothesis applies to φ_1 and φ_2 we have

  $$t \models \varphi_1 \wedge \varphi_2 \implies (t \models \varphi_1) \text{ and } (t \models \varphi_2) \implies$$
  $$(s \models \varphi_1) \text{ and } (s \models \varphi_2) \implies s \models \varphi_1 \wedge \varphi_2.$$

- $\varphi = \varphi_1 \vee \varphi_2$:

  Similar as for conjunction.

- $\varphi = \mathsf{EX}^{\mathsf{c}}_{\chi}\ \varphi_1$:

  If $t \models \varphi$ then there exists $t \xrightarrow{a,W} t'$, where $a \models \chi$, $t' \models \varphi_1$ and $\mathsf{c}(W) = 1$. Since $s \leq_m t$, we conclude that $s \xrightarrow{a,V} s'$, where $V \subseteq W$ exists such that $s' \leq_m t'$. The requirement on the constraint function c implies that also $\mathsf{c}(V) = 1$, since $V \subseteq W$. By the induction hypothesis $s' \models \varphi_1$ and thus $s \models \varphi$ as well.

- $\varphi = \mathsf{E}\ (\varphi_1\ \mathsf{U}^{\mathsf{c}}_{\chi}\ \varphi_2)$:

  If $t \models \varphi$ then there exists a must-path $\pi_N = t_1 \xrightarrow{a_1} t_2 \xrightarrow{a_2} \ldots$ with $t_1 = t$ in $\mathsf{maxmustP}(t)$ on which there exists j such that $t_j \models \varphi_2$, $t_k \models \varphi_1$ and $a_k \models \chi$ for all $k < j$ and $\mathsf{c}(\delta(t_1 \xrightarrow{a_1} t_2), \delta(t_2 \xrightarrow{a_2} t_3), \ldots, \delta(t_{j-1} \xrightarrow{a_{j-1}} t_j)) = 1$. By Lemma 11 we know that there exists a must-path $\pi_M = s_1 \xrightarrow{a_1} s_2 \xrightarrow{a_2} \ldots$ with $s_1 = s$ in $\mathsf{maxmustP}(s)$ such that $s_i \leq_m t_i$ for all i. The induction hypothesis implies that also $s_j \models \varphi_2$ and that $s_k \models \varphi_1$ for all $k < j$. By the requirement on c we also have that if $\mathsf{c}(\delta(t_1 \xrightarrow{a_1} t_2), \delta(t_2 \xrightarrow{a_2} t_3), \ldots, \delta(t_{j-1} \xrightarrow{a_{j-1}} t_j)) = 1$ then $\mathsf{c}(\delta(s_1 \xrightarrow{a_1} s_2), \delta(s_2 \xrightarrow{a_2} s_3), \ldots, \delta(s_{j-1} \xrightarrow{a_{j-1}} s_j)) = 1$, since $\delta(s_i \xrightarrow{a_i} s_{i+1}) \subseteq \delta(t_i \xrightarrow{a_i} t_{i+1})$ for all i. Hence $s \models \varphi$.

- $\varphi = \mathsf{E}\ (\varphi_1\ \mathsf{R}^{\mathsf{c}}_{\chi}\ \varphi_2)$:

  The same reasoning as in the previous case is used. As $t \models \varphi$ then there exists a must-path $\pi_N \in \mathsf{maxmustP}(t)$ such that φ_2 and χ holds in all states and transitions respectively until and including the first state where φ_1 holds (possibly never) and c equals 1 when evaluated on the weight intervals corresponding to the path where φ_2 holds. Again Lemma 11 provides a must-path $\pi_M \in \mathsf{maxmustP}(s)$ such that the induction hypothesis and the requirement on c gives us that $s \models \varphi$.

- $\varphi = \mathsf{AX}^{\mathsf{c}}_{\chi}\ \varphi_1$:

  Consider here an arbitrary transition $s \xrightarrow{a,V} s'$ where $a \models \chi$. Since $s \leq_m t$, there exists $t \xrightarrow{a,W} t'$ with $V \subseteq W$ such that $s' \leq_m t'$. As $t \models \varphi$, we know that $t' \models \varphi_1$ and $\mathsf{c}(W) = 1$. By the induction hypothesis and the requirement on c, we get that also $s \models \varphi$.

- $\varphi = \mathsf{A}\ (\varphi_1\ \mathsf{U}^{\mathsf{c}}_{\chi}\ \varphi_2)$:

  Consider an arbitrary path $\pi_M = s_1 \xrightarrow{a_1} s_2 \xrightarrow{a_2} \ldots$ with $s_1 = s$ in $\mathsf{maxmayP}(s)$ where $a_i \models \chi$ for all i. By Lemma 11 a path $\pi_N = t_1 \xrightarrow{a_1} t_2 \xrightarrow{a_2} \ldots$ with $t_1 = t$ in $\mathsf{maxmayP}(t)$ exists such that $s_i \leq_m t_i$ for all i. As $t \models \varphi$, then there exists j such that t_j satisfies φ_2, $t_k \models \varphi_1$ for all $k < j$ and $\mathsf{c}(\delta(t_1 \xrightarrow{a_1} t_2), \delta(t_2 \xrightarrow{a_2} t_3), \ldots, \delta(t_{j-1} \xrightarrow{a_{j-1}} t_j)) = 1$. By the induction hypothesis $s_j \models \varphi_2$ and $s_k \models \varphi_1$ for all $k < j$. Since $\delta(s_i \xrightarrow{a_i} s_{i+1}) \subseteq \delta(t_i \xrightarrow{a_i} t_{i+1})$ for all i, this implies that $\mathsf{c}(\delta(s_1 \xrightarrow{a_1} s_2), \delta(s_2 \xrightarrow{a_2} s_3), \ldots, \delta(t_{j-1} \xrightarrow{a_{j-1}} t_j)) = 1$. We now have that $s \models \varphi$ as required.

- $\varphi = \mathsf{A}\ (\varphi_1\ \mathsf{R}^{\mathsf{c}}_{\chi}\ \varphi_2)$:

  Again, we prove that if $t \models \varphi$ then also $s \models \varphi$. This is done as in the previous case by considering an arbitrary path in $\mathsf{maxmayP}(s)$ where all actions satisfy χ and applying Lemma 11, the induction hypothesis and the requirement on c. □

The reader may wonder why this action-based CTL is in positive normal form. The reason for this is that we require that a formula satisfied by a specification is also satisfied by all refinements. This does not hold for a formula of the form $\neg\varphi$. Consider for instance the specification consisting of two states s_1 and s_2, with $s_1 \xrightarrow{a,W} s_2$. Then the state s_1 satisfies $\varphi \equiv \neg\ \mathsf{EX}^{\mathsf{c}}_{a}\ \mathsf{true}$, with c returning constantly 1, since EX requires a must transition. However, in the refinement consisting of two states i_1 and i_2 with $i_1 \xrightarrow{a,V} i_2$, $V \subseteq W$, the state i_1 does not satisfy φ, since there indeed exists a must transition from s_1 with the action a. On the other hand, in the refinement consisting of an isolated state j_1 with no transitions $j_1 \models \varphi$ holds.

Thus, showing that a specification does not satisfy some φ only implies the existence of at least one refinement satisfying $\neg\varphi$, not all of them as required by the modal refinement methodology.

5. Conclusion and Future Work

We presented a novel extension of modal transition systems called weighted modal transition systems where each transition is decorated with an interval of weights, describing all possible values that can be used in an implementation. Furthermore we constructed the largest common refinement of a number of finite deterministic specifications, and proved the correctness of the construction. This result generalizes the previously known algorithm for the common implementation problem on unweighted deterministic modal transition systems. We also suggested a notion of weighted CTL logic in order to reason about the properties of the weighted modal transition systems and argued for the soundness of this choice.

Clearly the proposed logic is undecible due to its generality. As a future work it would therefore be interesting to identify decidable fragments of the logic. This can be achieved e.g. by considering a subset of the allowed state formulae, an unweighted version of the logic or by reasoning only about implementations.

In our future work we will also consider the common implementation/specification problems of nondeterministic specifications, a determinisation construction, algorithmic aspects of the generalized model checking problem for weighted modal specifications as well as the extension of the formalisms to mixed systems where the must transitions are not necessarily included in the may transitions. One might also consider a lattice of values as weight domain instead of the less general intervals considered here.

References

[1] K. G. Larsen, B. Thomsen, A Modal Process Logic, in: Proceedings of the 3rd Annual Symposium on Logic in Computer Science (LICS'88), IEEE Computer Society, 1988, pp. 203–210.

[2] J.-B. Raclet, Residual for Component Specifications, Electronic Notes in Theoretical Computer Science 215 (2008) 93–110.

[3] N. Bertrand, S. Pinchinat, J.-B. Raclet, Refinement and Consistency of Timed Modal Specifications, in: Proceedings of the 3rd International Conference on Language and Automata Theory and Applications (LATA'09), Vol. 5457 of LNCS, Springer-Verlag, 2009, pp. 152–163.

[4] S. Uchitel, M. Chechik, Merging partial behavioural models, in: Proceedings of the 12th ACM SIGSOFT International Symposium on Foundations of Software Engineerings (FSE'04), ACM, 2004, pp. 43–52.

[5] J.-B. Raclet, E. Badouel, A. Benveniste, B. Caillaud, R. Passerone, Why Are Modalities Good for Interface Theories?, in: Proceedings of the 9th International Conference on Application of Concurrency to System Design (ACSD'09), IEEE Computer Society, Los Alamitos, CA, USA, 2009, pp. 119–127.

[6] P. Godefroid, M. Huth, R. Jagadeesan, Abstraction-Based Model Checking Using Modal Transition Systems, in: Proceedings of the 12th International Conference on Concurrency Theory (CONCUR'01), Vol. 2154 of LNCS, Springer-Verlag, 2001, pp. 426–440.

[7] M. Huth, R. Jagadeesan, D. A. Schmidt, Modal Transition Systems: A Foundation for Three-Valued Program Analysis, in: Proceedings of the 10th European Symposium on Progamming (ESOP'01), Vol. 2028 of LNCS, Springer-Verlag, 2001, pp. 155–169.

[8] S. Nanz, F. Nielson, H. R. Nielson, Modal Abstractions of Concurrent Behaviour, in: Proceedings of 15th International Symposium on Static Analysis (SAS'08), Vol. 5079 of LNCS, Springer-Verlag, 2008, pp. 159–173.

[9] H. Fecher, H. Schmidt, Comparing Disjunctive Modal Transition Systems with an One-Selecting Variant, Journal of Logic and Algebraic Programming 77 (1-2) (2008) 20–39.

[10] O. Wei, A. Gurfinkel, M. Chechik, Mixed Transition Systems Revisited, in: Proceedings of the 10th International Conference on Verification, Model Checking and Abstract Interpretation (VMCAI'09), Vol. 5403 of LNCS, Springer-Verlag, 2009, pp. 349–365.

[11] A. Antonik, M. Huth, K. G. Larsen, U. Nyman, A. Wąsowski, 20 Years of Modal and Mixed Specifications, Bulletin of the EATCS (95) (2008) 94–129.

[12] A. Gruler, M. Leucker, K. D. Scheidemann, Modeling and Model Checking Software Product Lines, in: G. Barthe, F. S. de Boer (Eds.), FMOODS, Vol. 5051 of LNCS, Springer-Verlag, 2008, pp. 113–131.

[13] A. Gruler, M. Leucker, K. D. Scheidemann, Calculating and Modeling Common Parts of Software Product Lines, in: Proceedings of the 13th International Software Product Line Conference (SPLC'08), IEEE Computer Society, 2008, pp. 203–212.

[14] M. Droste, W. Kuich, H. Vogler (Eds.), Handbook of Weighted Automata, 1st Edition, Springer-Verlag, 2009.

[15] N. Beneš, J. Křetínský, K. G. Larsen, J. Srba, On determinism in modal transition systems, Theoretical Computer Science 410 (41) (2009) 4026–4043.

[16] T. Henzinger, J. Sifakis, The Embedded Systems Design Challenge, in: Proceedings of the 14th International Symposium on Formal Methods (FM'06), Vol. 4085 of LNCS, Springer-Verlag, 2006, pp. 1–15.

[17] T. A. Henzinger, J. Sifakis, The Discipline of Embedded Systems Design, IEEE Computer 40 (10) (2007) 32–40.

[18] R. D. Nicola, F. W. Vaandrager, Action versus State based Logics for Transition Systems, in: Semantics of Systems of Concurrent Processes, Vol. 469 of LNCS, Springer-Verlag, 1990, pp. 407–419.

[19] A. Lluch-Lafuente, U. Montanari, Quantitative μ-calculus and CTL defined over constraint semirings, Theoretical Computer Science 346 (1) (2005) 135–160.

[20] M. Droste, P. Gastin, Weighted Automata and Weighted Logics, in: Proceedings of the 7th International Colloquium on Automata, Languages and Programming (ICALP'05), Vol. 3580 of LNCS, Springer-Verlag, 2005, pp. 513–525.

[21] N. Beneš, J. Křetínský, K. G. Larsen, J. Srba, Checking Thorough Refinement on Modal Transition Systems Is EXPTIME-Complete, in: Proceedings of the 6th International Colloquium on Theoretical Aspects of Computing (ICTAC'09), Vol. 5684 of LNCS, Springer-Verlag, 2009, pp. 112–126.

[22] P. Kanellakis, S. Smolka, CCS Expressions, Finite State Processes, and Three Problems of Equivalence, Information and Computation 86 (1) (1990) 43–68.

[23] R. Paige, R. Tarjan, Three Partition Refinement Algorithms, SIAM Journal of Computing 16 (6) (1987) 973–989.

[24] E. A. Emerson, E. M. Clarke, Characterizing Correctness Properties of Parallel Programs Using Fixpoints, in: Proceedings of the 7th International Colloquium on Automata, Languages and Programming (ICALP'80), Vol. 85 of LNCS, Springer-Verlag, 1980, pp. 169–181.

[25] C. Baier, J.-P. Katoen, Principles of Model Checking, The MIT Press, 2008.

[26] R. Meolic, T. Kapus, Z. Brezocnik, ACTLW - An Action-Based Computation Tree Logic with Unless Operator, Information Sciences 178 (6) (2008) 1542–1557.

[27] P. Bouyer, U. Fahrenberg, K. G. Larsen, N. Markey, J. Srba, Infinite Runs in Weighted Timed Automata with Energy Constraints, in: F. Cassez, C. Jard (Eds.), Proceedings of the 6th International Conference on Formal Modelling and Analysis of Timed Systems (FORMATS'08), Vol. 5215 of LNCS, Springer-Verlag, Saint-Malo, France, 2008, pp. 33–47.

[28] G. Bruns, P. Godefroid, Generalized Model Checking: Reasoning about Partial State Spaces, in: C. Palamidessi (Ed.), Proceedings of the 11th International Conference on Concurrency Theory (CONCUR'00), Vol. 1877 of LNCS, Springer-Verlag, 2000, pp. 168–182.