Symbolic Synthesis of Non-Negative Multi-Weighted Games with Temporal Objectives

Master Thesis
DES102F17

Aalborg University
Department of Computer Science

Department of Computer Science
Aalborg University
http://www.aau.dk

AALBORG UNIVERSITY
STUDENT REPORT

Title:
Symbolic Synthesis of Non-Negative Multi-Weighted Games with Temporal Objectives

Theme:
Controller Synthesis

Project Period:
Spring Semester 2017

Project Group:
DES102F17

Participant(s):
Isabella Kaufmann
Lasse S. Jensen
Søren M. Nielsen

Supervisor(s):
Kim G. Larsen
Jiří Srba

Copies: 6

Page Numbers: 41

Date of Completion:
June 1, 2017

Abstract:

We provide a framework to describe non-negative multi-weighted Kripke structures and non-negative multi-weighted games. We investigate how to synthesize strategies for controller components with discrete weights using the framework. We show that model checking multi-weighted CTL over multi-weighted Kripke structures is undecidable with 3 weights and provide a decidable sub-logic, that can still express branching time properties. We also provide an algorithm utilizing a symbolic weight representation to synthesize finite memory strategies for multi-weighted games with reachability objectives and show that the problem is NP-hard and decidable in exponential time.

Preface

The project is the second part of our Master's Thesis from the Department of Computer Science at Aalborg University. It is written during the spring of 2017 and we would like to thank our supervisors Kim Guldstrand Larsen and Jiří Srba for the feedback they have given throughout the project.

Aalborg University, June 1, 2017

Isabella Kaufmann
<ikaufm12@student.aau.dk>

Lasse S. Jensen
<lasjen12@student.aau.dk>

Søren M. Nielsen
<smni12@student.aau.dk>

Symbolic Synthesis of Non-Negative Multi-Weighted Games with Temporal Objectives

Isabella Kaufmann, Lasse Steen Jensen & Søren Moss Nielsen

June 1, 2017

Summary

Synthesis is the automatic construction of software controllers from a formal specification. In contrast to human written software, which may exhibit undesirable behavior (E.g. deadlocks), such controllers can be guaranteed to behave according to its specification. In this thesis we investigate controller synthesis from a non-negative multi-weighted game theoretic context.

We start by introducing the underlying data structure of a game, which is a Kripke Strukture (KS) extended with multiple weights on each transition expressed as an n-dimensional vector of non-negative integers. We also define both syntax and semantics for a Weighted Computation Tree Logic (WCTL). The logic can express branching of time, comparison between weights and constants as well as arithmetics expressions. This allows us to specify multiple lower- and upper-bounds for each dimension of the vector. The model checking problem for WCTL over an n-weighted Kripke structure is examined and we find that with three dimensional weights, the problem is undecidable. We then introduce a sub-logic we call Constant Bound WCTL (cb-WCTL), where comparison between weights of the game has been removed from the logic and subtraction is not allowed as an arithmetic expression. We prove that the model checking problem for cb-WCTL on an n-weighted Kripke structure is decidable. Additionally we observe that the remaining arithmetic operators do not provide further expressiveness to the logic.

We introduce a game as a game graph and an objective. The game graph is presented as an n-weighted Kripke structure, where the transitions have been partitioned between two players and the objective is a formula expressed in a logic. In the context of controller synthesis we show how the first player corresponds to the controller and the second player corresponds to the environment around the controller. The game is played by the two players moving along the transitions in accordance to their ownership. The movements of the players are decided by a strategy. We formally define such a strategy with relation to the controller and present two types of strategies. In the first type of strategy all moves previously made are taken into account when choosing the next move, a so called full-memory strategy. In the second only the current state and the accumulation of the weights previously passed is considered, a so called finite-memory strategy. We illustrate that there exist objectives that can be satisfied, when the controller utilizes a full memory strategy, but cannot when utilizing a finite memory strategy.

In this thesis we focus on algorithmically solving the synthesis problem specified with reachability objectives. We first illustrate how to solve the synthesis problem, when the reachability objective only has upper-bounds defined. We do so by providing an algorithm and prove the correctness and complexity of the algorithm. We then consider games specified with lower-bounds as well and introduce the notion of zones to symbolically

represent the possible infinite set of weights of the game. We also introduce operations on these zones and prove them semantically equivalent to operations on infinite sets of weights. With these zones we provide an algorithm for solving the synthesis problem, where the reachability objective has both upper- and lower-bounds. We show correctness of this algorithm and show that while the problem is NP-hard the algorithm runs in exponential time.

Lastly we discuss the possibility of trivially extending the algorithm to handle non-nested cb-WCTL games. Additionally we show that the algorithm presented in this thesis is not capable of handling nested operators and suggest this as an area of future work.

1 Introduction

Controllers are used in everything from automated homes and self driving cars to manufacturing plants. These controllers often handle safety critical systems where errors can have severe consequences. One way of decreasing the amount of errors in these critical components is to use formal verification methods.

Controller synthesis is the automated construction of controllers derived directly from their specifications. The concept of controller synthesis was first introduced by Church in 1962 as a question of whether it was possible, algorithmically, to generate a solution for a specification formulated in a second-order monadic logic [5]. In 1969 Büchi and Landweber showed that this is possible [2]. Since then, the problem of controller synthesis has been continually expanded upon.

One way of working with synthesis is from a game theoretic view [18], focusing on non-cooperative multi-player (usually two player) games. Non-cooperative games are especially suited for synthesis of controllers, as these type of games do not necessarily assume an antagonistic opponent, although this is often the case, to ensure victory. The synthesis of a controller can be viewed as finding and extracting a winning strategy for one of these games.

In this thesis we work with two player games in a non-negative multi-weighted setting. A game is played on a game graph (also called an arena), which is a directed multi-weighted graph. A game graph consists of a set of states (vertices) and transitions (edges) where the transitions are partitioned between the players. Each transition has an associated weight expressed as a vector. When a game graph is paired with an objective we call it a game. A play of a game, is a sequence of states and transitions. To gain familiarity with the formalism, let us go through an example. Consider the game graph in Figure 1, which models a car insurance sales agent. States are depicted by circles, and transitions as arrows. The ordinary arrows depict the actions of the controller (sales agent) and the dashed arrows are the actions of the environment. The double circled state is the initial state, from which the game begins.

The game graph depicted in Figure 1 models three values. The first value *Risk* models the risk assessment of the costumer. The second value *Maximal_payout* models the maximal payout a costumer can receive when they make a claim. Lastly, the third value *Premium* models the price of the premium offered.

In the graph depicted in Figure 1 there are four states each illustrating a step in the sales process:

– First the customer is automatically assessed based on age, gender, insurance history etc. This is not done by the agent, so the action is chosen by the environment. We represent this by giving the initial state s_0 the label *Assess*. Progression to state s_1 is then along one of the dashed arrows.

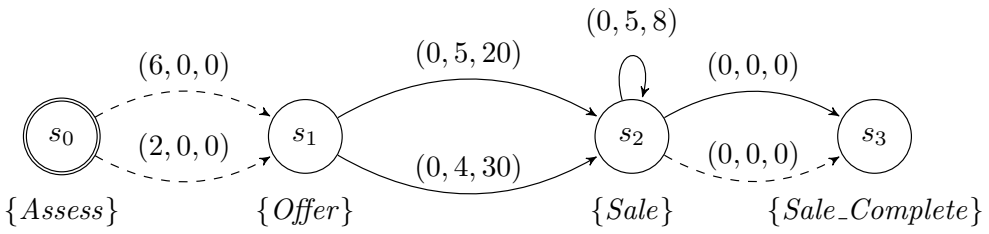


Figure 1: Example of a game graph modeling a car insurance sales agent. The three weight components represent the risk of the customer making a claim, the maximal payout of a claim and premium the sales agent can offer.

- In state s_1 with the label *Offer* the sales agent has two insurance policies available and he can offer either to the customer. These plans are represented by the two ordinary arrows from s_1 to s_2.
- After the offer is made the sales agent can optionally offer additional coverage along with a higher premium represented by the self-loop arrow. This can be declined by the customer represented by the dashed arrow. The decision to not offer further coverage, is represented by the ordinary arrow from s_2 to s_3.
- In the last state the sale has been made, represented by the label *Sale_Complete*.

Consider being the insurance company. You want to make sure that all customers pay a premium befitting their maximal payout compared, to the risk of the customer making a claim. Let us say we calculate the *total cost* of an insurance as $(Risk \cdot Maximal\_payout)$. We can then specify an instance of the problem as *"Can we offer a premium s.t. we can pay the costumer on every claim without loosing money?"*. You can now instruct your sales agents to follow a procedure that enforces these boundaries. An example of such a procedure is illustrated in Algorithm 1.

Algorithm 1 Sales agent strategy procedure

1: **if** $Offer \wedge Risk \leq 2$ **then goto** s_2 via the cost $(0, 5, 20)$
2: **else if** $Offer \wedge Risk > 2$ **then goto** s_2 via the cost $(0, 4, 30)$
3: **else if** $Sale \wedge Premium \leq 20$ **then goto** s_2
4: **else goto** s_3

Algorithm 1 is called a *strategy*. Consider an agent using this strategy, where the game starts with all dimensions of the weight set to 0 in the initial state s_0:

$s_0 - (0, 0, 0)$ The assessment can have two outcomes, either the customer is categorized as low risk or a high risk.

From this point there are two distinct strategies based on whether the costumer is low or high risk. Below we present the low risk strategy.

1. **$s_1 - (2, 0, 0)$** When the customer is categorized as low risk, the agent offers the policy with a maximal payout of 5 and a premium of 20.
2. **$s_2 - (2, 5, 20)$** With the premium at 20, the agent should offer further coverage to the customer. Mind that the customer may refuse this, thus there can be two outcomes.

One where the payout and premium is raised but the game stays in s_2 and one where the game progresses to s_3.

(a) $s_2 - (2, 10, 28)$ The customer has accepted further coverage, but the agent should not offer more, hence the game should progress to s_3.

3. $s_3 - (2, 5, 20)$ or $s_3 - (2, 10, 28)$ The sale is now closed by either the costumer or the agent and we observe that the total cost is either $2 \times 10 = 20$ while the premium is at 28 or $2 \times 5 = 10$ while the premuim is 20. Thus this strategy generates a profit for the insurance company.

Now we see that this strategy always turn a profit when the costumer is low risk. However, a different strategy is needed for a high risk costumer. This strategy is illustrated below.

$s_1 - (6, 0, 0)$ The strategy advises to offer the policy with a maximal payout of 4 and a premium of 30.

$s_2 - (6, 4, 30)$ As the costumer is high risk the strategy advises not to offer further coverage, and the model should progress to s_3.

$s_3 - (6, 4, 30)$ The sale is now closed and we observe that the total cost is $6 \times 4 = 24$ while the premium is at 30. Thus this strategy generates a profit for the insurance company.

Again we get that this ensures a profit, thus we have that no matter what the environment does, we can ensure that we satisfy our objective. This is what we call a *wining strategy* and it is the calculation and extraction of such strategies, we study in this thesis.

1.1 Contributions

In this thesis we present a game theoretic framework for two-player non-negative multi-weighted games. This framework is based on a weighted extension of Kripke Struktures (KSs) [12], the so called n-Weighted Kripke Strukture (n-WKS), and a weighted extension of Computation Tree Logic (CTL) [6], the Weighted Computation Tree Logic (WCTL). We provide an undecidability result for the model checking problem of WCTL on a 3-WKS and define the decidable sub-logic Constant Bound WCTL (cb-WCTL). To express reachability objectives with both lower and upper-bounds, we define a sub-logic of cb-WCTL, the Reachability WCTL with Upper- and Lower-bounds (ReachWCTL$_l^u$).

We show that the synthesis problem for a Reachability WCTL with Upper-bounds (ReachWCTL^u) game is decidable in pseudo-polynomial time. We provide a symbolic algorithm for the synthesis problem for ReachWCTL$_l^u$ games and prove semantically sound operations for a symbolic representation of the state space. We establish that the synthesis problem for ReachWCTL$_l^u$ games is NP-hard but decidable in exponential time and discuss the possibilities and challenges for synthesis of cb-WCTL. Furthermore, we show that if there is a winning strategy to a ReachWCTL^u or ReachWCTL$_l^u$ game, then there exists a winning strategy which only remembers the current state and the accumulated cost. Lastly, we show that the strategy needed for ReachWCTL^u and ReachWCTL$_l^u$ games does not suffice for cb-WCTL games.

1.2 Related Works

In [18] Thomas presented the *automata theoretic framework* defining a generalized two-player turn based game along with a strategy for that game. In [11] we extended this

framework to a non-negative multi-weighted setting, but only showed synthesis for one weighted games and only for upper-bounds. In this thesis, we use the framework from [11] in the multi-weighted setting but show synthesis of multi-weighted reachability games with both lower- and upper-bounds.

Inspired by the weighted logic of Jensen et al. in [8], we present a CTL variant with multiple weights. While Jensen et al. give a local algorithm for model checking of weighted CTL with only upper-bounds, we provide an algorithm for synthesis of multi-weighted reachability games defined with a reachability sub-logic with both lower- and upper-bounds. Bouyer, Larsen, and Markey [1] presented model checking for weighted timed automata, and showed that checking for weighted CTL properties with three or more clocks is undecidable. We arrive at the same result for WCTL with three weights, and define the decidable sub-logic cb-WCTL.

Kupferman and Vardi found in [14], that the synthesis problem for CTL is EXPTIME complete. We show that the decidable sub-logic of WCTL is NP-hard and multi-weighted reachability objectives with upper- and lower-bounds can be solved in exponential time. Furthermore, Kupferman and Vardi show that synthesis of CTL objectives in games with partial information is 2EXPTIME complete. In our thesis we only consider games with complete information.

Mean pay-off games and energy games are suitable for modeling resource constrained non-terminating systems. In 2015 Jurdziński, Lazić, and Schmitz presented an algorithm for solving fixed-dimensional energy games in pseudo-polynomial time [10]. In the games we study we can only model consumption or accumulation of some resource, as all weights are positive. Chatterjee et al. showed that the problem of determining if a finite memory strategy exists for energy games is co-NP complete and NP-complete if the strategy is memoryless [4]. We show that games with reachability objectives only require a finite memory strategy.

In 2005 Cassez et al. presented an on-the-fly approach for solving both untimed and timed automaton based games with reachability and safety objectives using dependency graphs by Liu and Smolka in [15]. Timed Automata (TA) based games model continuous times, while our games model multiple discrete values by using a symbolic abstraction. Additionally our method is based on n-WKS with reachability with upper- and lower-bounds objectives.

In [9] Jobstmann and Bloem present the tool Lily for synthesis of LTL specifications. It uses a translation through universal co-Büchi tree automata and alternating weak tree automata first presented by Kupferman and Vardi in [13]. Lily does not consider weights, as opposed to our framework.

1.3 Outline

In Section 2 we introduce the preliminaries including the n-WKS and WCTL. Section 3 cover the model checking results for WCTL and cb-WCTL while Section 4 present the synthesis problem in a game theoretic context. In Section 5 we present an algorithmic solution for the synthesis problem for reachability objectives and we finish with a discussion on synthesis of cb-WCTL in Section 6. Lastly, we conclude on our results in Section 7.

2 Preliminaries

We present the basic formalism and notation used throughout the thesis. We define the n-WKS, a KS with arbitrary many non-negative weights on the transitions. Then we define the syntax and semantics of WCTL over an n-WKS.

2.1 n-Weighted Kripke Strukture

We present the n-Weighted Kripke Strukture (n-WKS), and we write $\mathbb{N}_0 = \mathbb{N} \cup \{0\}$ and $\mathbb{N}_\infty = \mathbb{N}_0 \cup \infty$.

Definition 2.1 (n-Weighted Kripke Strukture)
An n-WKS is a tuple $K = (S, s_0, \mathcal{AP}, L, T)$ where:

- S is a set of states,
- $s_0 \in S$ is the initial state,
- $\mathcal{AP}$ is a finite set of atomic propositions,
- $L : S \to \mathcal{P}(\mathcal{AP})$ is a labeling function and
- $T \subseteq S \times \mathbb{N}_0^n \times S$ is a transition relation, with a weight vector of n dimensions.

When $(s, \overline{c}, s') \in T$, where $s, s' \in S$ and $\overline{c} \in \mathbb{N}_0^n$ is a vector, then we write $s \xrightarrow{\overline{c}} s'$. When s' is reachable from s, by $j \in \mathbb{N}$ number of transitions, we write $s \to^j s'$ and when s has no outgoing transitions we write $s \not\to$.

An n-WKS $K = (S, s_0, \mathcal{AP}, L, T)$ is *finite* whenever S is a finite set of states and T is a finite transition relation.

Let $\overline{w} \in \mathbb{N}_0^n$ then we denote the ith component of $\overline{w}$ by $\overline{w}[i]$, where $1 \leq i \leq n$. To set the ith component of $\overline{w}$ to a specific value $k \in \mathbb{N}_0$ we write $\overline{w}[i \to k]$.

Definition 2.2 (Ordering on Vectors)
Let $\overline{w} = (\overline{w}[1], \ldots, \overline{w}[n]) \in \mathbb{N}_0^n$ and $\overline{w}' = (\overline{w}'[1], \ldots, \overline{w}'[n]) \in \mathbb{N}_0^n$. We write $\overline{w} \leq \overline{w}'$ iff $\overline{w}[i] \leq \overline{w}'[i]$ for all $1 \leq i \leq n$.

We define a run ρ in the n-WKS K to be an infinite or finite sequence of states and transitions:

$$\rho = s_1 \xrightarrow{\overline{c}_1} s_2 \xrightarrow{\overline{c}_2} s_3 \xrightarrow{\overline{c}_3} \ldots$$

where $s_i \xrightarrow{\overline{c}_i} s_{i+1}$ for all $i \geq 1$. Given a position $i \in \mathbb{N}$ along ρ, let $\rho(i) = s_i$, and $\textsc{Last}(\rho)$ be the state at last position along ρ, if ρ is finite. We also define the concatenation operator $\circ$, s.t. if $(\rho = s_1 \xrightarrow{\overline{c}_1} s_2 \xrightarrow{\overline{c}_2} \ldots \xrightarrow{\overline{c}_{n-1}} s_n)$ then $\rho \circ (s_n \xrightarrow{\overline{c}_n} s_{n+1}) = (s_1 \xrightarrow{\overline{c}_1} s_2 \xrightarrow{\overline{c}_2} s_3 \ldots s_n \xrightarrow{\overline{c}_n} s_{n+1})$.

We write the set of all runs ρ in the n-WKS K of the form $(\rho = s_1 \xrightarrow{\overline{c}_1} s_2 \xrightarrow{\overline{c}_2} \ldots)$ as Π_K. Furthermore we write the set of all finite runs ρ in the n-WKS K of the form $(\rho = s_1 \xrightarrow{\overline{c}_1} \ldots \xrightarrow{\overline{c}_{n-1}} s_n)$ as Π_K^{fin}. Lastly, we define Π_K^{Max} as the set of all runs ρ s.t. ρ is infinite or $\textsc{Last}(\rho)$ is in a deadlock s.t. $\textsc{Last}(\rho) \not\to$.

Definition 2.3 (Cost)
Let $K = (S, s_0, \mathcal{AP}, L, T)$ be an n-WKS and $(\rho = s_1 \xrightarrow{\overline{c}_1} s_2 \xrightarrow{\overline{c}_2} \ldots)$ be a run in K. The cost of ρ, at position $i \in \mathbb{N}$, is then defined as:

$$\textsc{Cost}_\rho(i) = \begin{cases} 0^n & \text{if } i = 1 \\ \sum_{j=1}^{i-1} \overline{c}_j & \text{otherwise.} \end{cases}$$

If ρ is finite, we denote $\textsc{Cost}(\rho)$ as the cost of the last position along ρ.

2.2 Weighted Computation Tree Logic

We define Weighted Computation Tree Logic (WCTL) in relation to an n-WKS $K = (S, s_0, \mathcal{AP}, L, T)$ s.t.

$$\begin{aligned}\varphi :=& \textsc{true} \mid \textsc{false} \mid a \mid \psi_1 \bowtie \psi_2 \mid \neg\varphi \mid \\ & \varphi_1 \vee \varphi_2 \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \Rightarrow \varphi_2 \mid \varphi_1 \Leftrightarrow \varphi_2 \mid \\ & AX\ \varphi \mid EX\ \varphi \mid AG\ \varphi \mid EG\ \varphi \mid AF\ \varphi \mid EF\ \varphi \mid \\ & E\varphi_1 U \varphi_2 \mid A\varphi_1 U \varphi_2 \mid \textit{reset } \#i \text{ in } \varphi \\ \psi :=& c \mid \#i \mid \psi_1 \oplus \psi_2\end{aligned}$$

where $a \in \mathcal{AP}$, $\bowtie \in \{<, \leq, =, \geq, >\}$, $\oplus \in \{+, -, \cdot\}$, $c \in \mathbb{N}_0$, and $1 \leq i \leq n$ is a component index in a vector. Given a WCTL formula φ, we define the set of all sub-formulae in φ as $Sub(\varphi)$.

We define the semantics for a minimal set of operators. Let $s \in S$ be a state and $\overline{w} \in \mathbb{N}_0^n$. We then write $K, s \vDash_{\overline{w}} \varphi$ when K satisfies the formula φ in the state s with the cost $\overline{w}$.

$K, s \vDash_{\overline{w}} \textsc{true}$

$K, s \vDash_{\overline{w}} a$ iff $a \in L(s)$

$K, s \vDash_{\overline{w}} \neg\varphi$ iff $s \nvDash_{\overline{w}} \varphi$

$K, s \vDash_{\overline{w}} \varphi_1 \vee \varphi_2$ iff $s \vDash_{\overline{w}} \varphi_1$ or $s \vDash_{\overline{w}} \varphi_2$

$K, s \vDash_{\overline{w}} E\varphi_1 U \varphi_2$ iff there exists $(\rho = s_1 \xrightarrow{\bar{c}_1} s_2 \xrightarrow{\bar{c}_2} s_3 \dots) \in \Pi_K^{Max}$ where $s = s_1$ and a position $i \geq 1$ such that $K, \rho(i) \vDash_{\overline{w}+cost_\rho(i)} \varphi_2$ and $K, \rho(j) \vDash_{\overline{w}+cost_\rho(j)} \varphi_1$ for all $j < i$

$K, s \vDash_{\overline{w}} A\varphi_1 U \varphi_2$ iff for all $(\rho = s_1 \xrightarrow{\bar{c}_1} s_2 \xrightarrow{\bar{c}_2} s_3 \dots) \in \Pi_K^{Max}$ where $s = s_1$, there is a position $i \geq 1$ such that $K, \rho(i) \vDash_{\overline{w}+cost_\rho(i)} \varphi_2$ and $K, \rho(j) \vDash_{\overline{w}+cost_\rho(j)} \varphi_1$ for all $j < i$

$K, s \vDash_{\overline{w}} EX\ \varphi$ iff there is a state s' such that $s \xrightarrow{\bar{c}} s'$, and $K, s' \vDash_{\overline{w}+\bar{c}} \varphi$

$K, s \vDash_{\overline{w}} \textit{reset } \#i \text{ in } \varphi$ iff $K, s \vDash_{\overline{w}[i \to 0]} \varphi$

$K, s \vDash_{\overline{w}} \psi_1 \bowtie \psi_2$ iff $eval_{\overline{w}}(\psi_1) \bowtie eval_{\overline{w}}(\psi_2)$

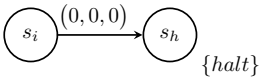
The evaluation of ψ is:

$$\begin{aligned} eval_{\overline{w}}(c) &= c \\ eval_{\overline{w}}(\#i) &= \overline{w}[i] \\ eval_{\overline{w}}(e_1 \oplus e_2) &= eval_{\overline{w}}(e_1) \oplus eval_{\overline{w}}(e_2) \end{aligned}$$

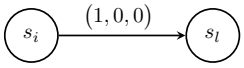
The remaining operators, from the syntax, can be derived from the minimal set, and likewise can their semantics. The derived operators are defined as:

$$\begin{aligned} AF\ \varphi &\equiv A(\textsc{true})U(\varphi) & EF\ \varphi &\equiv E(\textsc{true})U(\varphi) \\ AG\ \varphi &\equiv \neg EF \neg\varphi & EG\ \varphi &\equiv \neg AF \neg\varphi \\ AX\ \varphi &\equiv \neg EX(\neg\varphi) & \varphi_1 \wedge \varphi_2 &\equiv \neg(\neg\varphi_1 \vee \neg\varphi_2) \\ \varphi_1 \Rightarrow \varphi_2 &\equiv \neg\varphi_1 \vee \varphi_2 & \varphi_1 \Leftrightarrow \varphi_2 &\equiv (\varphi_1 \Rightarrow \varphi_2) \wedge (\varphi_2 \Rightarrow \varphi_1) \end{aligned}$$

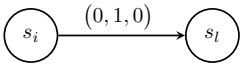
Given an n-WKS $K = (S, s_0, \mathcal{AP}, L, T)$ and a WCTL formula φ the model checking problem is the question of whether $K, s \vDash_{\overline{w}} \varphi$ where $s \in S$ and $\overline{w} \in \mathbb{N}_0^n$. When the n-WKS K is obvious from context and the vector $\overline{w}$ is 0^n we simply write $s \vDash \varphi$.



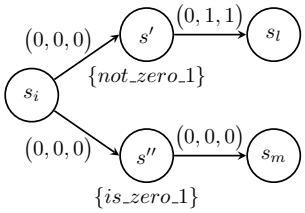
(a) Halt rule.



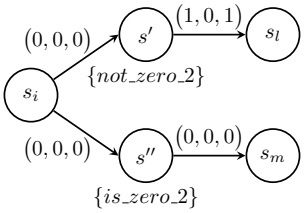
(b) Increment rule for C_1.



(c) Increment rule for C_2.



(d) Decrement rule for C_1.



(e) Decrement rule for C_2.

Figure 2: n-WKS simulation of a 2CM

3 Model checking

In this section we look at decidability of the model checking problem for WCTL. First we show that the model checking problem for WCTL is undecidable on a finite 3-WKS. We then present a sub-logic and prove it is decidable.

Theorem 1 (Undecidability of WCTL)
The model checking problem for WCTL is undecidable on a finite 3-WKS.

Proof We use reduction from the halting problem for 2-counter Minsky machines [16]. Let M be a two-counter-machine (2CM) with two non-negative counters C_1 and C_2 and a finite set of instructions where each instruction Ins_i is either

- e: HALT
- Increment i: $C_j := C_j + 1$; GOTO(l)
- Decrement i: If $C_j > 0$ then $(C_j := C_j - 1$; GOTO(l)) else GOTO(m)

where $j \in (1,2)$ and $1 \leq l, m \leq e$. To simulate the machine, let K be a finite 3-WKS where whenever K is in state s_i then M is in Ins_i. We use the vector of length three to increase and decrease the value of the counters, by the cost of a run $\rho \in \Pi_K^{fin}$ s.t.

$$C_1 = \text{COST}_\rho(i)[1] - \text{COST}_\rho(i)[3],$$

$$C_2 = \text{COST}_\rho(i)[2] - \text{COST}_\rho(i)[3],$$

where $0 \leq i$ is the position of s_i in the run ρ. In the simulation, the instructions are translated as seen in Figure 2 s.t. HALT is simulated in Figure 2a, Increment in Figure 2b and 2c and Decrement in Figure 2d and Figure 2e.

We then construct the formula,

$$\varphi = E(A \wedge B)U(halt)$$

where

$$A := (not\_zero\_1) \Rightarrow \#1 > \#3 \wedge (not\_zero\_2) \Rightarrow \#2 > \#3$$
$$B := (is\_zero\_1) \Rightarrow \#1 = \#3 \wedge (is\_zero\_2) \Rightarrow \#2 = \#3$$

We now want to show that M will halt for the empty input $(C_1 = 0, C_2 = 0)$ iff $s_0 \vDash \varphi$.

- $\Rightarrow$ If M halts, then $K, s_0 \vDash_{0^n} \varphi$. We know that when running M the n-WKS K can simulate the exact same instructions, so that if M will halt, then a state s_h is reached in K by a run ρ s.t. $\textsc{Last}(\rho) = s_h$ and for all $0 \leq i$ s.t. $\rho(i) \neq \textsc{Last}(\rho)$ it holds that $K, \rho(i) \vDash A \wedge B$, hence if M halts, then $K, s_0 \vDash_{0^n} \varphi$.
- $\Leftarrow$ If $K, s_0 \vDash_{0^n} \varphi$ then M will halt. The formula φ ensures that M is simulated faithfully, as the counters are calculated in φ. When encountering the decrement rules, φ enforces that the correct choice is taken, as the next state of the path will never satisfy both pre-conditions A and B if it is not allowed in M. To simulate the empty input, the weight vector $\overline{w} = 0^n$. As M is faithfully simulated, we have that if $K, s_0 \vDash_{0^n} \varphi$ then M will halt.

Hence M will halt for the empty input $(C_1 = 0, C_2 = 0)$ iff $s_0 \vDash \varphi$ and therefore we can conclude that WCTL is undecidable on a finite 3-WKS. □

3.1 Constant Bound WCTL

We observe that the ability to simulate a 2-counter Minsky machine makes the model checking problem for WCTL undecidable. Specifically we note that the comparison of vector components and the use of subtraction between vector components makes it possible to model the counters and the zero check. Based on this observation we define the Constant Bound WCTL (cb-WCTL) without comparison of vector components and subtraction.

Let $K = (S, s_0, \mathcal{AP}, L, T)$ be an n-WKS. The cb-WCTL syntax is then defined over K as follows,

$$\begin{aligned}
\varphi :=& \textsc{true} \mid \textsc{false} \mid a \mid \psi \bowtie c \mid \neg\varphi \mid \varphi_1 \vee \varphi_2 \mid \\
& \varphi_1 \wedge \varphi_2 \mid \varphi_1 \Rightarrow \varphi_2 \mid \varphi_1 \Leftrightarrow \varphi_2 \mid \mathit{reset}\ \#i \text{ in } \varphi \mid \\
& AX\ \varphi \mid EX\ \varphi \mid AG\ \varphi \mid EG\ \varphi \mid AF\ \varphi \mid \\
& EF\ \varphi \mid E\varphi_1 U \varphi_2 \mid A\varphi_1 U \varphi_2 \\
\psi :=& \#i \mid c \mid \psi_1 \oplus \psi_2
\end{aligned}$$

where $a \in \mathcal{AP}$, $\bowtie \in \{<, \leq, =, \geq, >\}$, $\oplus \in \{+, \cdot\}$, $c \in \mathbb{N}_0$, and $1 \leq i \leq n$ is a component index in a vector.

We define Φ as the set of all cb-WCTL formulae. Notice that $\oplus$ is now restricted to addition and multiplication, and that the right hand-side of $\bowtie$ is now restricted to a constant. Thus where we had $\psi_1 \bowtie \psi_2$ in WCTL, we now have $\psi \bowtie c$ in cb-WCTL.

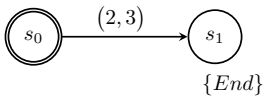
We argue that while we allow addition and multiplication in the logic, it does not add any expressiveness to the logic. Let $K = (S, s_0, \mathcal{AP}, L, T)$ be an n-WKS and φ a cb-WCTL formula. We say that a sub-formula $\psi \bowtie c$ is *atomic* if and only if $\psi = \#i$ or $\psi = c$, and *non-atomic* otherwise. To illustrate the lack of added expressiveness we show that

any non-atomic proposition can be expressed as an atomic proposition and an additional weight in the n-WKS.

First we isolate each sub-formula, that is not already atomic s.t. given a formula φ, we replace all non-atomic $\psi \bowtie c \in Sub(\varphi)$ with $\#(n+1) \bowtie c$ where $\#(n+1)$ is a new component in the vector and we modify K as follows:

$$\text{for all } s \xrightarrow{\overline{w}=(w_1,w_2,\ldots,w_n)} s' \text{ in } K, s \xrightarrow{\overline{w'}=(w_1,w_2,\ldots,w_n,eval_{\overline{w}}(\psi))} s' \text{ in } K'$$

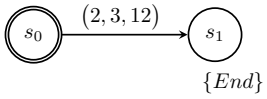
Creating a cb-WCTL formula φ' and computing the corresponding n-WKS K' can now be done using this simple procedure. The resulting model checking problem $K', s_0 \vDash \varphi'$ will have the same answer as the original problem $K, s_0 \vDash \varphi$. Consider the model checking problem $K, s_0 \vDash \varphi$ where K is the n-WKS shown in Figure 3a and φ is the cb-WCTL formula shown in Figure 3b.



(a) n-WKS K

$$AF(End \wedge \#1 \geq 1 \wedge (\#1 \times (\#2 + 3)) \leq 15)$$

(b) cb-WCTL formula φ



(c) n-WKS K'

$$AF(End \wedge \#1 \geq 1 \wedge \#3 \leq 15))$$

(d) cb-WCTL formula φ'

Figure 3: Transformation of the n-WKS K and cb-WCTL formula φ

Creating the corresponding problem $K', s_0 \vDash \varphi'$ where φ' is a cb-WCTL formula is done by replacing the sub-formula $\#1 \times (\#2 + 3)$ with the component #3 and adding a third weight which is equal to the evaluated expression. This is illustrated in Figure 3c and Figure 3d.

3.1.1 Decidability of cb-WCTL

We will now show that cb-WCTL is decidable on a finite n-WKS. We prove decidability for the model checking problem of cb-WCTL on a finite n-WKS, by reducing it to the model checking problem for CTL on a finite KS which is decidable [6]. To do so we utilize the fact that the cost of a run is non-decreasing and bounds are specified by some positive constant. From this we observe that at some point the cost of a run no longer affect the satisfiability of a cb-WCTL formula. Therefore we can make a finite unfolding of an n-WKS and translate the weights into atomic propositions. As shown earlier, we can, without loss of generality, assume that a formula does not contain any addition or multiplication operators.

Let $K = (S, s_0, \mathcal{AP}, L, T)$ be an n-WKS and φ be a cb-WCTL formula. Recall that $Sub(\varphi)$ is the set of all sub-formula in φ. We then define the boundary of φ as a vector $\overline{b}$ where for all dimension $1 \leq i \leq n$:

$$\overline{b}[i] = \begin{cases} \max\{c \mid (\#i \bowtie c) \in Sub(\varphi)\} & \text{if there exist } (\#i \bowtie c) \in Sub(\varphi) \\ -1 & \text{otherwise} \end{cases}$$

Notice, if $\overline{b}[i] = -1$, for some $i \in \mathbb{N}_0$, then there is no bound (upper or lower) on that dimension of the game in φ. We say the boundary $\overline{b}$ is *derived from* φ. With this we can define a function used to limit the amount of vectors represented in K, allowing us to finitely unfold K. We call this function Cut and define it as:

Definition 3.1 (Cut)
Given an n-WKS $K = (S, s_0, \mathcal{AP}, L, T)$, a cb-WCTL formula φ and let $\overline{b}$ be the bound derived from φ. We then define the function $Cut : \mathbb{N}_0^n \to \mathbb{N}_0^n$ s.t. for all $1 \leq i \leq n$:

$$Cut(\overline{w})[i] = \begin{cases} \overline{w}[i] & \text{if } \overline{w}[i] \leq \overline{b}[i] \\ \overline{b}[i] + 1 & \text{otherwise} \end{cases}.$$

Lemma 2
Given an n-WKS $K = (S, s_0, \mathcal{AP}, L, T)$, a cb-WCTL formula φ and a vector $\overline{w} \in \mathbb{N}_0^n$, it holds that $K, s \vDash_{\overline{w}} \varphi$ iff. $K, s \vDash_{Cut(\overline{w})} \varphi$

Proof We will now prove that $K, s \vDash_{\overline{w}} \varphi$ iff. $K, s \vDash_{Cut(\overline{w})} \varphi$ by structural induction on φ.

$\varphi = \#i \leq c$:

- $\Rightarrow$: Assume $s \vDash_{\overline{w}} \#i \leq c$. Then by the definition of Cut we have that $Cut(\overline{w})[i] = \overline{w}[i] \leq c$. Thus $s \vDash_{Cut(\overline{w})} \#i \leq c$.
- $\Leftarrow$: Assume $s \vDash_{Cut(\overline{w})} \#i \leq c$. Then by the definition of Cut we have that $\overline{w}[i] = Cut(\overline{w})[i] \leq c$. Thus $s \vDash_{\overline{w}} \#i \leq c$.

$\varphi = \#i \geq c$:

- $\Rightarrow$: Assume $s \vDash_{\overline{w}} \#i \geq c$. Then by the definition of Cut we have that $\overline{w}[i] \geq Cut(\overline{w})[i] \geq c$, since $\overline{b}[i] \geq c$ where $\overline{b}$ is derived from φ. Thus $s \vDash_{Cut(\overline{w})} \#i \geq c$.
- $\Leftarrow$: Assume $s \vDash_{Cut(\overline{w})} \#i \geq c$. Then by the definition of Cut we have that $\overline{w}[i] \geq Cut(\overline{w})[i] \geq c$. Thus $s \vDash_{\overline{w}} \#i \geq c$.

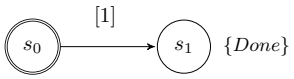
The remaining cases, in both directions, follow trivially from the induction hypothesis, since no other operators depend on the weights. Thus we can conclude that $K, s \vDash_{\overline{w}} \varphi$ iff. $K, s \vDash_{Cut(\overline{w})} \varphi$. □

Next, we define how to finitely unfold $K = (S, s_0, \mathcal{AP}, L, T)$ into a KS $K' = (S', (s_0^0, 0^n), \mathcal{AP}', L', T')$ and from there show that the problem can be reduced to classical CTL model checking. Intuitively we create a state $(s, \overline{w}) \in S'$ for each state $s \in S$ paired with a vector in the co-domain of Cut, and give them the labels, that were originally assigned to s. In order to convert the weight evaluation to CTL, we add the labels $(\#i \bowtie c)$ to the new state, if the weight of the new state satisfies $\#i \bowtie c$. Additionally, we create intermediate states to handle the reset operator. These *reset states* are named with super script notation to indicate what component index is to be reset, where super script 0 indicates that the state is not a reset state. For example, let $(s^i, \overline{w})$, where $i > 0$, be a reset state for $(s^0, \overline{w}) \in S'$. Now $(s^i, \overline{w})$ acts as an intermediate state between $(s^0, \overline{w})$ and $(s^0, \overline{w}[i \to 0]) \in S'$, such that $(s^0, \overline{w}) \to (s^i, \overline{w}) \to (s^0, \overline{w}[i \to 0])$. We then assign the atomic proposition $reset_i$ to $(s^i, \overline{w})$ allowing us to model the resetting of the ith coordinate. To differentiate between reset states and non-reset states, the label *not_reset* is given to all 0 indexed states. This unfolding is formally defined in Definition 3.2.

Definition 3.2 (Finite unfolding of an n-WKS)
Given the model checking problem for $K, s_0 \vDash_{0^n} \varphi$ where $K = (S, s_0, \mathcal{AP}, L, T)$ is an n-WKS and φ a cb-WCTL formula, we construct the KS $K' = (S', (s_0^0, 0^n), \mathcal{AP}', L', T')$ where

- $S' = \{(s^i, Cut(\overline{w})) \mid s \in S \text{ and } \overline{w} \in \mathbb{N}_0^n \text{ and } 0 \leq i \leq n\}$.
- $(s_0^0, 0^n) \in S'$ is the initial state.
- $\mathcal{AP}' = \mathcal{AP} \cup \{(\#i \bowtie c) \mid (\#i \bowtie c) \in Sub(\varphi)\} \cup \{reset_i \mid 1 \leq i \leq n\} \cup \{not\_reset\}$.
- For any $(s^0, \overline{w}) \in S'$, $L'((s^0, \overline{w}))$ is defined as:
  - $\alpha \in L'((s^0, \overline{w}))$ iff. $\alpha \in L(s)$ for any $\alpha \in \mathcal{AP}$,
  - $(\#i \bowtie c) \in L'((s^0, \overline{w}))$ iff. $\overline{w}[i] \bowtie c$ for any $1 \leq i \leq n$ where $(\#i \bowtie c) \in Sub(\varphi)$,
  - $not\_reset \in L'((s^0, \overline{w}))$, and
  - for any $i \in \mathbb{N}$, $reset_i \notin L'((s^0, \overline{w}))$.
- For any $(s^i, \overline{w}) \in S'$ where $1 \leq i \leq n$, $L'((s^i, \overline{w}))$ is defined as:
  - $L'((s^i, \overline{w})) = \{reset_i\}$.
- T' is defined, for any $(s^i, \overline{w}) \in S'$ where $0 \leq i \leq n$, as:
  - $(s^0, \overline{w}) \to (s'^0, Cut(\overline{w} + \overline{c})) \in T'$ iff. $s \xrightarrow{\overline{c}} s' \in T$, and
  - $(s^0, \overline{w}) \to (s^i, \overline{w}), (s^i, \overline{w}) \to (s^0, \overline{w}[i \to 0]) \in T'$ for any $1 \leq i \leq n$.

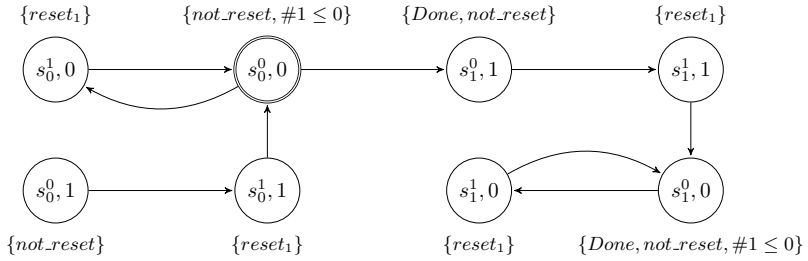
An example of the unfolding is illustrated in Figure 4 where Figure 4a shows the original n-WKS K, Figure 4b shows the cb-WCTL formula φ and Figure 4c shows the unfolded KS K'. Notice how the reset states are only given the label $reset_1$. While all other states are given the labels the state had in the original n-WKS, the not_reset label and the label of the weight comparison they satisfy.



(a) 1-WKS $K = (S, s_0, \mathcal{AP}, L, T)$

$$\varphi = E\,(\text{TRUE})\; U\; (reset\ \#1\ \text{in}\ (Done \wedge \#1 \leq 0))$$

(b) cb-WCTL formula φ



(c) KS $K' = (S', (s_0^0, 0), \mathcal{AP}', L', T')$ unfolded from K.

Figure 4: An 1-WKS K unfolded into a KS K' using the method in Definition 3.2, where the query φ is used to determine the maximal value that Cut can return.

Since cb-WCTL contains operators unknown in classical CTL we show how to translate a cb-WCTL formula into a CTL formula, by converting the weight related operators. We assume a standard definition of the CTL syntax and semantics. To differentiate between the evaluation of cb-WCTL formulas and CTL we write $K, s \overset{ctl}{\models} \varphi$ where K is a KS, s is a state of K and φ is a CTL formula. We define Ψ as the set of all CTL formulae.

Definition 3.3 (CTL Translation function)
Let φ be a cb-WCTL formula. We define a function for translating a cb-WCTL formula into a CTL formula, called $ctl : \Phi \to \Psi$ as:

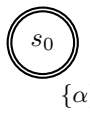
$$ctl(\varphi) = \begin{cases} \alpha & \text{if } \varphi = \alpha \in \mathcal{AP} \\ \text{TRUE} & \text{if } \varphi = \text{TRUE} \\ (\#i \bowtie c) & \text{if } \varphi = \#i \bowtie c \\ \neg ctl(\varphi') & \text{if } \varphi = \neg\varphi' \\ ctl(\varphi_1) \vee ctl(\varphi_2) & \text{if } \varphi = \varphi_1 \vee \varphi_2 \\ EX\ (ctl(\varphi_1) \wedge not\_reset) & \text{if } \varphi = EX\ \varphi_1 \\ EX\ (reset_i\ \wedge EX\ (ctl(\varphi'))) & \text{if } \varphi = reset\ i\ in\ \varphi' \\ E(ctl(\varphi_1) \wedge not\_reset) U (ctl(\varphi_2) \wedge not\_reset) & \text{if } \varphi = E\varphi_1 U \varphi_2 \\ A \begin{pmatrix} ctl(\varphi_1) \wedge \\ not\_reset \wedge \\ EX\ not\_reset \end{pmatrix} U (ctl(\varphi_2)\ \vee \neg not\_reset) & \text{if } \varphi = A\varphi_1 U \varphi_2 \end{cases}$$

The remaining operators of cb-WCTL are translated according to the operators they are derived from, specified in Section 2.2.

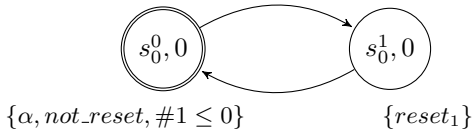
Notice that the case for $A\varphi_1 U \varphi_2$ is significantly different in $ctl(\varphi)$ than in φ. There are two reasons for this.

- Because the KS constructed in Definition 3.2 has transitions to the reset states, there now exist runs that do not correspond to any runs in the n-WKS it was constructed from. In order to disregard these branches when testing all paths, we make sure that the until clause is satisfied as they are reached.
- Because the KS constructed in Definition 3.2, will no longer deadlock, where the original n-WKS would we need to test for liveness. Consider Figure 5, the n-WKS K in Figure 5a clearly has a deadlock and the KS K' constructed from K in Figure 5b is live. In Figure 5d the cb-WCTL formula φ is illustrated above the CTL translation of φ. Obviously $K, s_0 \not\models_0 \varphi$ in Figure 5c, since there is no label β in K, but as we have added reset states in K' where we accept the until condition ($\beta\ \vee \neg not\_reset$), we must test if the state was live in the original n-WKS. We test for liveness with the $EX(not\_reset)$ condition in $ctl(\varphi)$, otherwise $K', (s_0^0, 0) \overset{ctl}{\models} ctl(\varphi)$.

Now we simply need to prove that the model checking problem for a cb-WCTL formula φ on a n-WKS K is equivalent to the model checking problem of the translated formula $ctl(\varphi)$ on the unfolded KS K'. To do so we define a few semantic equivalences necessary in the proof. We begin by presenting a distance based semantic of the $A\psi_1 U \psi_2$ formula.



(a) $K = (S, s_0, \mathcal{AP}, L, T)$.



(b) KS $K' = (S', (s_0^0, 0^n), \mathcal{AP}', L', T')$.

$$\varphi = A\left(\alpha \wedge \#1 \leq 0\right) U\left(\beta\right)$$

(c) cb-WCTL formula.

$$ctl(\varphi) = A\begin{pmatrix}\alpha \wedge (\#1 \leq 0) \wedge \\ not\_reset \wedge \\ EX\ not\_reset\end{pmatrix} U\ \beta\ \vee \neg not\_reset$$

(d) The CTL formula $ctl(\varphi)$

Figure 5: An n-WKS where there is a deadlock, but there is none in the unfolded KS.

Definition 3.4 (Distance based AU semantics)
Let $K = (S, s_0, \mathcal{AP}, L, T)$ be an n-WKS. For any $s \in S$ and any $\overline{w} \in \mathbb{N}_0^n$ we define that $K, s \vDash_{\overline{w}} A\ \varphi_1\ U^j\ \varphi_2$ for some $j \in \mathbb{N}_0$ s.t.

- $s \vDash_{\overline{w}} A\ \varphi_1\ U^0\ \varphi_2$ iff. $s \vDash_{\overline{w}} \varphi_2$, and
- $s \vDash_{\overline{w}} A\ \varphi_1\ U^j\ \varphi_2$ iff. $s \vDash_{\overline{w}} \varphi_1$ and $j = \max\{j \mid s \xrightarrow{\overline{c}} s' \wedge s' \vDash_{\overline{w}+\overline{c}} A\ \varphi_1\ U^{j-1}\ \varphi_2\}$.

Obviously, we have a similar construction for $ctl(A\varphi_1 U \varphi_2)$.

Based on this definition we derive the following proposition.

Proposition 1
Let $K = (S, s_0, \mathcal{AP}, L, T)$ be an n-WKS. For any $s \in S$ and any $\overline{w} \in \mathbb{N}_0^n$, then it holds that $s \vDash_{\overline{w}} A\ \varphi_1\ U\ \varphi_2$ iff. $s \vDash_{\overline{w}} A\ \varphi_1\ U^j\ \varphi_2$ for some $j \in \mathbb{N}_0$. Obviously, it also holds that given any $(s, Cut(\overline{w})) \in S'$ and $\overline{w} \in \mathbb{N}_0^n$ then $(s, Cut(\overline{w})) \overset{ctl}{\vDash} ctl(A\ \varphi_1\ U\ \varphi_2)$ iff. $(s, Cut(\overline{w})) \overset{ctl}{\vDash} ctl(A\ \varphi_1\ U^j\ \varphi_2)$ for some $j \in \mathbb{N}_0$.

To further decompose the $A\psi_1 U \psi_2$ formula, we present an unfolding of the formula based on the distance.

Proposition 2
By semantic equivalence we have that

$$A\varphi_1 U^j \varphi_2 \equiv \varphi_2 \vee (\varphi_1 \wedge EX(\textsc{true}) \wedge AX\ (A\varphi_1 U^{j-1} \varphi_2)).$$

Proof Let $K = (S, s_0, \mathcal{AP}, L, T)$ be an n-WKS. Assume for some $s \in S$ and some $\overline{w} \in \mathbb{N}_0^n$ that $s \vDash_{\overline{w}} A\varphi_1 U^j \varphi_2$ then by Proposition 1 $s \vDash_{\overline{w}} \varphi_2$, or $s \vDash_{\overline{w}} \varphi_1$ and $j = \max\{j \mid s \xrightarrow{\overline{c}} s' \wedge s' \vDash_{\overline{w}+\overline{c}} A\ \varphi_1\ U^{j-1}\ \varphi_2\}$. Clearly, either $s \vDash_{\overline{w}} \varphi_2$, or $s \vDash_{\overline{w}} \varphi_1$ and there exist $s \xrightarrow{\overline{c}} s' \in T$ and for all $s \xrightarrow{\overline{c}} s' \in T$ it holds that $s' \vDash_{\overline{w}+\overline{c}} A\ \varphi_1\ U^{j-1}\ \varphi_2$. Hence $A\varphi_1 U^j \varphi_2 \equiv \varphi_2 \vee (\varphi_1 \wedge EX(\textsc{true}) \wedge AX\ (A\varphi_1 U^{j-1} \varphi_2))$. □

Lemma 3
Given an n-WKS $K = (S, s_0, \mathcal{AP}, L, T)$ and a cb-WCTL formula φ then let $K' = (S', (s_0^0, 0^n), \mathcal{AP}', L', T')$ be a KS constructed from Definition 3.2. For any $s \in S$ and any $\overline{w} \in \mathbb{N}_0$ it holds that $K', (s^0, Cut(\overline{w})) \overset{ctl}{\vDash} ctl(\varphi)$ iff. $K, s \vDash_{\overline{w}} \varphi$.

Proof We will now prove that $K', (s^0, Cut(\overline{w})) \models^{ctl} ctl(\varphi)$ iff. $K, s \models_{\overline{w}} \varphi$. The proof is by structural induction on the formula φ.

$\varphi = \#i \bowtie c$:

$\Rightarrow$: Assume $(s^0, Cut(\overline{w})) \models^{ctl} ctl(\#i \bowtie c)$, then $(\#i \bowtie c) \in L'((s^0, Cut(\overline{w})))$. By the construction of K' we have that $(\#i \bowtie c) \in L'((s^0, Cut(\overline{w})))$ iff. $Cut(\overline{w})[i] \bowtie c$, thus $K, s \models_{Cut(\overline{w})} \#i \bowtie c$. By Lemma 2 we know that $K, s \models_{Cut(\overline{w})} \varphi$ iff. $K, s \models_{\overline{w}} \varphi$. Hence $K, s \models_{\overline{w}} \varphi$.

$\Leftarrow$: Assume $K, s \models_{\overline{w}} \#i \bowtie c$, and by Lemma 2 we know that $s \models_{\overline{w}} \#i \bowtie c$ iff. $s \models_{Cut(\overline{w})} \#i \bowtie c$. Thus by the construction of K' there must exist $(s^0, Cut(\overline{w})) \in S'$ where $(\#i \bowtie c) \in L((s^0, Cut(\overline{w})))$. Hence $K', (s^0, Cut(\overline{w})) \models^{ctl} ctl(\#i \bowtie c)$.

$\varphi = reset\ \#i\ in\ \varphi'$:

$\Rightarrow$: Assume $(s^0, Cut(\overline{w})) \models^{ctl} EX(reset_i \wedge EX(ctl(\varphi')))$. By the construction of K' we know that only $(s^i, Cut(\overline{w})) \models^{ctl} reset_i$ and that $(s^i, Cut(\overline{w}))$ only has a single outgoing transition $(s^i, Cut(\overline{w})) \rightarrow (s^0, Cut(\overline{w}[i \rightarrow 0]) \in T'$. Thus $(s^0, Cut(\overline{w}[i \rightarrow 0]) \models^{ctl} \varphi'$ and by the induction hypothesis we have that $s \models_{Cut(\overline{w}[i \rightarrow 0])} \varphi'$. Then by the semantics $s \models_{Cut(\overline{w})} reset\ \#i\ in\ \varphi'$, hence by Lemma 2 it must hold that $K, s \models_{\overline{w}} \varphi$.

$\Leftarrow$: Assume $K, s \models_{\overline{w}} \varphi$, then $s \models_{\overline{w}[i \rightarrow 0]} \varphi'$ and by Lemma 2 we know that $s \models_{Cut(\overline{w}[i \rightarrow 0])} \varphi'$. By the induction hypothesis we have that $(s^0, Cut(\overline{w}[i \rightarrow 0])) \models^{ctl} \varphi'$. By the construction of K' there exist $(s^0, Cut(\overline{w})), (s^i, Cut(\overline{w})) \in S'$ s.t. $(s^0, Cut(\overline{w})) \rightarrow (s^i, Cut(\overline{w})) \rightarrow (s^0, Cut(\overline{w}[i \rightarrow 0]))$ where $(s^i, Cut(\overline{w})) \models^{ctl} reset_i$. Thus $(s^0, Cut(\overline{w})) \models^{ctl} EX\ (reset_i\ \wedge EX\ ctl(\varphi'))$, hence $(s^0, Cut(\overline{w})) \models^{ctl} ctl(reset\ \#i\ in\ \varphi')$.

$\varphi = A\ \varphi_1\ U\ \varphi_2$:

$\Rightarrow$: Assume $(s^0, Cut(\overline{w})) \models^{ctl} A(ctl(\varphi_1) \wedge not\_reset \wedge EX(not\_reset) U (ctl(\varphi_2) \vee \neg not\_reset)$. By Proposition 1 we then have that $(s^0, Cut(\overline{w})) \models^{ctl} ctl(A\ \varphi_1\ U^j\ \varphi_2)$ for some $j \in \mathbb{N}_0$. By induction on j, we prove that if $(s^0, Cut(\overline{w})) \models^{ctl} ctl(A\ \varphi_1\ U^j\ \varphi_2)$ then $K, s \models_{\overline{w}} A\ \varphi_1\ U\ \varphi_2$:

Base: Let $j = 0$. Then by Definition 3.4 $(s^0, Cut(\overline{w})) \models^{ctl} \varphi_2 \vee \neg not\_reset$. By the construction of K', $not\_reset \in L'((s^0, Cut(\overline{w})))$, hence $K', (s^0, Cut(\overline{w})) \models^{ctl} \varphi_2$ and by the structural induction hypothesis $K, s \models_{Cut(\overline{w})} \varphi_2$. Thus by the semantics it holds that $K, s \models_{Cut(\overline{w})} A\varphi_1 U \varphi_2$ and by Lemma 2 that $K, s \models_{\overline{w}} A\varphi_1 U \varphi_2$.

Induction step: Let $j > 0$. Assume $(s^0, Cut(\overline{w})) \models^{ctl} ctl(A\ \varphi_1\ U^j\ \varphi_2)$. By Proposition 2 $(s^0, Cut(\overline{w})) \models^{ctl} ctl(\varphi_2 \vee (\varphi_1 \wedge EX(\text{TRUE}) \wedge AX(A\varphi_1 U^{j-1} \varphi_2)))$. Since $j > 0$ we have that $(s^0, Cut(\overline{w})) \not\models^{ctl} ctl(\varphi_2)$, and hence $(s^0, Cut(\overline{w})) \models^{ctl} ctl(\varphi_1 \wedge EX(\text{TRUE}) \wedge AX(A\varphi_1 U^{j-1} \varphi_2))$. By the semantics $(s^0, Cut(\overline{w}))$ satisfies each clause of the conjunctions, and we analyze each clause individually:

$(s^0, Cut(\overline{w})) \models^{ctl} ctl(\varphi_1)$: By the structural induction hypothesis $K, s \models_{Cut(\overline{w})} \varphi_1$ and by Lemma 2 then $K, s \models_{\overline{w}} \varphi_1$.

$(s^0, Cut(\overline{w})) \models^{ctl} ctl(EX(\text{true}))$: By the translation of $ctl(EX(\varphi'))$, we have that $(s^0, Cut(\overline{w})) \models^{ctl} EX(\text{TRUE} \wedge not\_reset)$, hence there exists $(s^0, Cut(\overline{w})) \rightarrow (s'^0, Cut(\overline{w}')) \in T'$ s.t. $not\_reset \in L'((s'^0, Cut(\overline{w}')))$. By construction of K' there must exist $s \xrightarrow{\overline{c}} s' \in T$ where $\overline{w}' = \overline{w} + \overline{c}$ hence by the semantics $K, s \models_{\overline{w}} EX(\text{TRUE})$.

$(s^0, Cut(\overline{w})) \models^{ctl} ctl(AX(A\varphi_1 U^{j-1}\varphi_2))$: This implies that for all $(s^0, Cut(\overline{w})) \rightarrow (s'^0, Cut(\overline{w}')) \in T'$ it holds that $(s'^0, Cut(\overline{w}')) \models^{ctl} ctl(A\varphi_1 U^{j-1}\varphi_2)$ and by the induction hypothesis we have for all $s \xrightarrow{\overline{c}} s' \in T$ that $K, s' \models_{\overline{w}+\overline{c}} A\varphi_1 U\varphi_2$, and that $\overline{w}' = \overline{w} + \overline{c}$. Hence by the semantics $K, s \models_{\overline{w}} AX(A\varphi_1 U\varphi_2)$.

This means that $K, s \models_{\overline{w}} \varphi_1 \wedge EX(\text{TRUE}) \wedge AX(A\varphi_1 U\varphi_2)$ when $j > 0$, thus by Proposition 2 we have that $K, s \models_{\overline{w}} A\varphi_1 U\varphi_2$ when $(s^0, Cut(\overline{w})) \models^{ctl} ctl(A\ \varphi_1\ U^j\ \varphi_2)$ for some $j > 0$.

Hence by induction on j, we have that if $(s^0, Cut(\overline{w})) \models^{ctl} ctl(A\ \varphi_1\ U^j\ \varphi_2)$ then $K, s \models_{\overline{w}} A\ \varphi_1\ U\ \varphi_2$. Thus by Proposition 1 if $(s^0, Cut(\overline{w})) \models^{ctl} ctl(A\ \varphi_1\ U\ \varphi_2)$ then $K, s \models_{\overline{w}} A\ \varphi_1\ U\ \varphi_2$.

$\Leftarrow$: Assume $K, s \models_{\overline{w}} A\ \varphi_1\ U\ \varphi_2$. By Proposition 1 we then have that $K, s \models_{\overline{w}} A\varphi_1 U^j \varphi_2$ for some $j \in \mathbb{N}_0$. By induction on j we prove that if $K, s \models_{\overline{w}} A\varphi_1 U^j \varphi_2$ then $(s^0, Cut(\overline{w})) \models^{ctl} ctl(A\ \varphi_1\ U\ \varphi_2)$.

Base: Let $j = 0$. Then by Definition 3.4 $K, s \models_{\overline{w}} \varphi_2$. By the structural induction hypothesis it holds that $(s^0, Cut(\overline{w})) \models^{ctl} ctl(\varphi_2)$, hence $(s^0, Cut(\overline{w})) \models^{ctl} ctl(\varphi_2) \vee \neg not\_reset$. Then by the semantics $(s^0, Cut(\overline{w})) \models^{ctl} ctl(A\varphi_1 U\varphi_2)$.

Induction step: Let $j > 0$. Assume $K, s \models_{\overline{w}} A\varphi_1 U^j \varphi_2$. Then by Proposition 2 $K, s \models_{\overline{w}} \varphi_2 \vee (\varphi_1 \wedge EX(\text{TRUE}) \wedge AX(A\varphi_1 U^{j-1}\varphi_2))$. Since $j > 0$ then $K, s \not\models_{\overline{w}} \varphi_2$ thus $K, s \models_{\overline{w}} \varphi_1 \wedge EX(\text{TRUE}) \wedge AX(A\varphi_1 U^{j-1}\varphi_2)$. By the semantics s satisfies each clause of the conjunctions, and we analyze each clause individually:

$K, s \models_{\overline{w}} \varphi_1$: By the structural induction hypothesis this implies that $(s^0, Cut(\overline{w})) \models^{ctl} ctl(\varphi_1)$.

$K, s \models_{\overline{w}} EX(\text{true})$: This implies that $s \xrightarrow{\overline{c}} s' \in T$ and by the construction of K' there must then exist $(s^0, Cut(\overline{w})) \rightarrow (s'^0, Cut(\overline{w} + \overline{c})) \in T'$ s.t. $not\_reset \in L'((s'^0, Cut(\overline{w} + \overline{c})))$, hence by the semantics $(s^0, Cut(\overline{w})) \models^{ctl} EX(\text{TRUE} \wedge not\_reset)$ and thus $(s^0, Cut(\overline{w})) \models^{ctl} ctl(EX(\text{TRUE}))$.

$K, s \models_{\overline{w}} AX(A\varphi_1 U^{j-1}\varphi_2)$: This means that for all $s \xrightarrow{\overline{c}} s' \in T$ it holds that $K, s' \models_{\overline{w}+\overline{c}} A\varphi_1 U^{j-1}\varphi_2$ and by the induction hypothesis it holds that for all $(s^0, Cut(\overline{w})) \rightarrow (s'^0, Cut(\overline{w} + \overline{c})) \in T'$ that $(s'^0, Cut(\overline{w} + \overline{c})) \models^{ctl} ctl(A\ \varphi_1\ U\ \varphi_2)$. Now, for any $(s^0, Cut(\overline{w})) \rightarrow (s^i, Cut(\overline{w})) \in T'$, where $i > 0$, it holds trivially that $(s^i, Cut(\overline{w})) \models^{ctl} ctl(A\ \varphi_1\ U\ \varphi_2)$, since by construction of K', $not\_reset \notin L'((s^i, Cut(\overline{w})))$. Hence $(s^0, Cut(\overline{w})) \models^{ctl} ctl(AX(A\varphi_1 U\varphi_2))$.

Thus by induction on j it holds that if $K, s \models_{\overline{w}} A\varphi_1 U^j \varphi_2$ then $(s^0, Cut(\overline{w})) \models^{ctl} ctl(\varphi_1) \wedge ctl(EX(\text{TRUE})) \wedge ctl(AX(A\varphi_1 U\varphi_2))$. This means that $(s^0, Cut(\overline{w})) \models^{ctl} ctl(\varphi_1 \wedge EX(\text{TRUE}) \wedge AX(A\varphi_1 U\varphi_2))$ and hence by Proposition 2 we have that $(s^0, Cut(\overline{w})) \models^{ctl} ctl(A\varphi_1 U\varphi_2)$, when $j > 0$.

We can therefore conclude that if $K, s \models_{\overline{w}} A\varphi_1 U^j \varphi_2$ for some $j \in \mathbb{N}_0$ then $(s^0, Cut(\overline{w})) \models^{ctl} ctl(A\ \varphi_1\ U\ \varphi_2)$ and by Proposition 1 we have that if $K, s \models_{\overline{w}} A\varphi_1 U\varphi_2$ then $(s^0, Cut(\overline{w})) \models^{ctl} ctl(A\ \varphi_1\ U\ \varphi_2)$.

Hence $(s^0, Cut(\overline{w})) \models^{ctl} ctl(A\ \varphi_1\ U\ \varphi_2)$ iff. $K, s \models_{\overline{w}} A\ \varphi_1\ U\ \varphi_2$.

$\varphi = E\varphi_1 U\varphi_2$:

$\Rightarrow$: Assume $(s^0, Cut(\overline{w})) \models^{ctl} E(ctl(\varphi_1) \wedge not\_reset)U(ctl(\varphi_2) \wedge not\_reset)$. Then there exists a run $(s^0, Cut(\overline{w})) \rightarrow^* (s_i^0, Cut(\overline{w}'))$ in K' s.t. $(s_i^0, Cut(\overline{w}')) \models^{ctl} ctl(\varphi_2) \wedge$

not_reset where $0 \leq i$. By the structural induction hypothesis $K, s_i \vDash_{Cut(\overline{w}')} \varphi_2$ and by Lemma 2 $K, s_i \vDash_{\overline{w}'} \varphi_2$. If $s^0 = s_i^0$, we are done. If $s^0 \neq s_i^0$ then it also holds for all $0 \leq j < i$ that $(s_j^0, Cut(\overline{w}_j)) \models^{ctl} ctl(\varphi_1) \wedge not\_reset$ and by the structural induction hypothesis $K, s_j \vDash_{Cut(\overline{w}_j)} \varphi_1$. By the construction of K' there exists a run $\rho = (s \xrightarrow{\overline{w}_1} s_1 \rightarrow^* s_i)$ in K, where we by the structural induction hypothesis have that $K, s_j \vDash_{Cut(\overline{w}_j)} \varphi_1$ for all $0 \leq j < i$ and that $K, s_i \vDash_{Cut(\overline{w}_i)} \varphi_2$, where $\overline{w}_j, \overline{w}_i \in \mathbb{N}_0^n$. Thus by the semantics $K, s \vDash_{Cut(\overline{w})} E\varphi_1 U \varphi_2$ and by Lemma 2 we have that $K, s \vDash_{\overline{w}} E\varphi_1 U \varphi_2$.

$\Leftarrow$: Assume $s \vDash_{\overline{w}} E\varphi_1 U \varphi_2$. Then there exists $(\rho = s \xrightarrow{\bar{c}_0} s_1 \xrightarrow{\bar{c}_1} s_2 \dots) \in \Pi_K^{Max}$ and a position $0 \leq i$ such that $K, \rho(i) \vDash_{\overline{w}+cost_\rho(i)} \varphi_2$ and $K, \rho(j) \vDash_{\overline{w}+cost_\rho(j)} \varphi_1$ for all $j < i$. Then by the structural induction hypothesis there exists $s_i = \rho(i)$ s.t. $(s_i^0, Cut(\overline{w} + cost_\rho(i))) \models^{ctl} ctl(\varphi_2)$ and by the construction of K' we have that $not\_reset \in L'((s_i^0, Cut(\overline{w} + cost_\rho(i))))$, hence $(s_i^0, Cut(\overline{w} + cost_\rho(i))) \models^{ctl} ctl(\varphi_2) \wedge not\_reset$. If $s = \rho(i)$ then $(s^0, Cut(\overline{w})) \models^{ctl} ctl(\varphi_2) \wedge not\_reset$, hence $(s^0, Cut(\overline{w})) \models^{ctl} ctl(E\varphi_1 U \varphi_2)$.

If $s \neq \rho(i)$ then by the structural induction hypothesis $(s_j^0, Cut(\overline{w} + cost_\rho(j))) \models^{ctl} ctl(\varphi_1)$ for all $j < i$, where $s_j = \rho(j)$. By the construction of K' we also have that for all $j < i$, $not\_reset \in L'((s_j^0, Cut(\overline{w} + cost_\rho(j))))$ and that all these form a run in K'. Thus for each $(s_j^0, Cut(\overline{w} + cost_\rho(j)))$ along the run it holds that $(s_j^0, Cut(\overline{w} + cost_\rho(j))) \models^{ctl} ctl(\varphi_1) \wedge not\_reset$ and by the semantics $(s^0, Cut(\overline{w})) \models^{ctl} ctl(E\varphi_1 U \varphi_2)$.

$\varphi = EX\ \varphi'$:

$\Rightarrow$: Assume $(s^0, Cut(\overline{w})) \models^{ctl} EX(ctl(\varphi') \wedge not\_reset)$, then $(s^0, Cut(\overline{w})) \rightarrow (s'^0, Cut(\overline{w}')) \in T'$, where $(s'^0, Cut(\overline{w}')) \models^{ctl} \varphi'$ and $not\_reset \in L'((s'^0, Cut(\overline{w}')))$. By the construction of K', T' is defined such that $(s \xrightarrow{\bar{c}} s') \in T$, and for that s' it holds that $s' \vDash_{Cut(\overline{w}')} \varphi'$, where $\overline{w}' = \overline{w} + \bar{c}$ by the structural induction hypothesis. From the semantics of cb-WCTL we then have that $s \vDash_{Cut(\overline{w})} EX\ \varphi'$ and thus by Lemma 2 it holds that $s \vDash_{\overline{w}} EX\ \varphi'$.

$\Leftarrow$: Assume $s \vDash_{\overline{w}} EX\ \varphi'$. Then by the semantics of cb-WCTL, there exists $(s \xrightarrow{\bar{c}} s') \in T$ such that $s' \vDash_{\overline{w}+\bar{c}} \varphi'$. By Lemma 2 we have that $s' \vDash_{Cut(\overline{w}+\bar{c})} \varphi'$ and by the construction of K' there exists $(s'^0, Cut(\overline{w} + \bar{c})) \in S'$ such that $not\_reset \in L'((s'^0, Cut(\overline{w} + \bar{c})))$ and there exists $(s^0, Cut(\overline{w})) \rightarrow (s'^0, Cut(\overline{w} + \bar{c})) \in T'$ where $(s^0, Cut(\overline{w})) \in S'$. By the induction hypothesis we then have that $(s'^0, Cut(\overline{w} + \bar{c})) \models^{ctl} \varphi'$, thus $(s^0, Cut(\overline{w})) \models^{ctl} EX\varphi'$ and by construction $(s'^0, Cut(\overline{w} + \bar{c})) \models^{ctl} not\_reset$, hence $(s^0, Cut(\overline{w})) \models^{ctl} ctl(EX\varphi')$.

Since each operator in cb-WCTL derive from these operators, we can conclude that $K', (s^0, Cut(\overline{w})) \models^{ctl} ctl(\varphi)$ iff. $K, s \vDash_{\overline{w}} \varphi$. □

Theorem 4 (Decidability of cb-WCTL)
The model checking problem for cb-WCTL is decidable on a finite n-WKS K.

Proof By Lemma 3 we may construct a finite KS K' from K such that $K', (s, Cut(\overline{w})) \models^{ctl} ctl(\varphi)$ iff. $K, s \vDash_{\overline{w}} \varphi$. Since the model checking problem for CTL is decidable on a finite KS, the model checking problem on K is also decidable for cb-WCTL. □

4 Games & Strategies

In this section we present a framework for non-negated multi-weighted games. We begin by presenting the notion of an n-Weighted Game (n-WG) and a strategy. We then present two distinct types of strategies and make observations on their expensiveness and limitations. Lastly we present the synthesis problem and show how n-WGs and the model checking problem of cb-WCTL on a n-WKS relate to controller synthesis.

4.1 Game

An n-WG is a two-player game where one player acts as the controller and the other player acts as the environment. A game is played on a game graph, where the transitions have been partitioned between the two players. When we pair a game graph with an objective, we call it an n-WG.

Definition 4.1 (n-Weighted Game Graph)
An n-Weighted Game Graph is a tuple $\mathcal{G} = (S, s_0, \mathcal{AP}, L, T_c, T_u)$ where T_c and T_u are disjoint sets and $K = (S, s_0, \mathcal{AP}, L, T_c \cup T_u)$ is an n-WKS.

The underlying data structure of the game graph is an n-WKS and we denote the specific n-WKS for the game graph $\mathcal{G} = (S, s_0, \mathcal{AP}, L, T_c, T_u)$ as $K_{\mathcal{G}} = (S, s_0, \mathcal{AP}, L, T_c \cup T_u)$. The set of transitions T_c is owned by the controller, and the set T_u is owned by the environment. We write:

$$s \xrightarrow{\overline{c}} s' \text{ if } (s, \overline{c}, s') \in T_c$$

$$s \overset{\overline{c}}{\dashrightarrow} s' \text{ if } (s, \overline{c}, s') \in T_u$$

From here on we will refer to transitions of the type $\rightarrow$ as controllable transitions and $\dashrightarrow$ as uncontrollable transitions. We write $s \rightarrow$ when there is some controllable outgoing transitions from s and $s \dashrightarrow$ when there is some uncontrollable outgoing transition from s.

Lastly we define a winning condition (objective) as a cb-WCTL formula φ over the n-WKS $K_{\mathcal{G}} = (S, s_0, \mathcal{AP}, L, T_c \cup T_u)$.

Definition 4.2 (Game)
Given an n-Weighted Game Graph (n-WGG) $\mathcal{G}$ and a cb-WCTL formula φ as a winning condition. We define the resulting tuple $(\mathcal{G}, \varphi)$ as a *game*.

The game starts in the initial state s_0. It progresses from state to state, through the choices of a player. A player can only choose to advance the game, by picking an outgoing transition belonging to her. The target state of that transition then becomes the current state.

- If all outgoing transitions belong to T_c, then the controller must choose.
- If all outgoing transitions belong to T_u, then the environment must choose.
- If there are both outgoing transitions in T_c and T_u, then the environment may choose one from T_u or force the controller to choose from T_c.

We give precedence to the environment when there are both controllable and uncontrollable outgoing transitions as this enforces the notion of an environment beyond our control.

4.2 Strategy

A strategy defines the player's actions, that is which transition to choose in which state of the game. As we want to model systems where the controller needs to be ready for anything the environment does, we limit our concern to that of the controller. This means that a strategy in the context of this thesis is a collection of predetermined transitions the controller will choose when playing the game, given any progression of the game. Thus when a run in the game progresses to a state where there is an outgoing controllable transition, a strategy will return the chosen transition of the controller.

Recall that $\Pi^{fin}_{K_\mathcal{G}}$ is the set of all finite runs in $K_\mathcal{G}$. We now define a strategy as a function mapping from finite runs to transitions.

Definition 4.3 (Strategy)
Let $\mathcal{G} = (S, s_0, \mathcal{AP}, L, T_c, T_u)$ be an n-WGG, then the strategy of the controller is a function $\sigma : \Pi^{fin}_{K_\mathcal{G}} \to T_c \cup \{\textsc{nil}\}$ mapping a finite run ρ to a transition s.t.

$$\sigma(\rho) = \begin{cases} \textsc{Last}(\rho) \xrightarrow{\bar{c}} s' \in T_c & \text{if } \textsc{Last}(\rho) \to \\ \textsc{nil} & \text{else} \end{cases}$$

and NIL is the choice to do nothing.

Given a strategy σ, we can apply that strategy onto the n-WGG $\mathcal{G}$ and get a modified n-WKS, by only keeping the controllable transitions that are in the co-domain of σ.

Definition 4.4 (Strategy restricted n-WKS)
Given a game graph $\mathcal{G} = (S, s_0, \mathcal{AP}, L, T_c, T_u)$ and a strategy σ, we define $\mathcal{G}\restriction\sigma = (S', s_0, \mathcal{AP}, L', T_c\restriction\sigma \cup T'_u)$ as the n-WKS resulting from restricting the game graph under the strategy σ.

- $S' = \Pi^{fin}_{K_\mathcal{G}}$
- $L'(\rho) = L(\textsc{Last}(\rho))$
- $T'_c\restriction\sigma = \{(\rho, \bar{c}, (\rho \circ \sigma(\rho)) \mid \sigma(\rho) = (\textsc{Last}(\rho) \xrightarrow{\bar{c}} s') \in T_c\}$
- $T'_u = \{(\rho, \bar{c}, \rho \circ (\textsc{Last}(\rho) \overset{\bar{c}}{\dashrightarrow} s')) \mid (\textsc{Last}(\rho) \overset{\bar{c}}{\dashrightarrow} s') \in T_u\}$.

In future illustrations we only include the part of $\mathcal{G}\restriction\sigma$ reachable from s_0. We say that a strategy is a *winning* if the n-WKS, resulting from restricting the game graph by the strategy, satisfy the winning condition.

Definition 4.5 (Winning Strategy)
The strategy σ is a *winning strategy* over the game $(\mathcal{G}, \varphi)$ iff $\mathcal{G}\restriction\sigma, s_0 \vDash_{0^n} \varphi$, where $\mathcal{G} = (S, s_0, \mathcal{AP}, L, T_c, T_u)$ and φ is a cb-WCTL formula.

To show the unfolding defined in Definition 4.4 and how this correlates with a winning strategy we encourage the reader to go through Example 4.1.

Example 4.1 (Strategy unfolding)
Consider the game $(\mathcal{G}, \varphi)$ where $\mathcal{G}$ is illustrated in Figure 6 and $\varphi = A\ (a)\ U\ ((b \wedge \#2 \geq 8) \vee (b \wedge \#1 \leq 2))$ is a cb-WCTL formula.

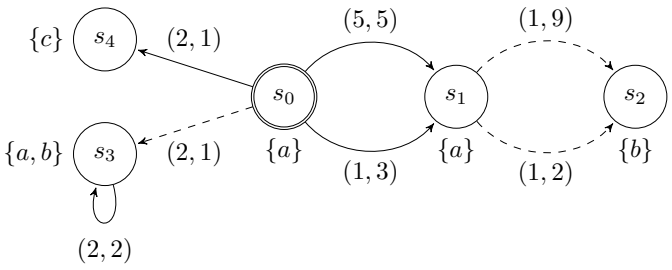


Figure 6: n-WGG $\mathcal{G} = (S, s_0, \mathcal{AP}, L, T_c, T_u)$.

For this game we can define a winning strategy σ s.t. for all $\rho \in \Pi^{fin}_{K_\mathcal{G}}$ then,

$$\sigma(\rho) = \begin{cases} s_0 \xrightarrow{(1,3)} s_1 & \text{if } \textsc{Last}(\rho) = s_0 \\ s_3 \xrightarrow{(2,2)} s_3 & \text{if } \textsc{Last}(\rho) = s_3 \\ \textsc{nil} & \text{if } \textsc{Last}(\rho) \not\to \end{cases}$$

Given this strategy we get the strategy restricted n-WKS $\mathcal{G}\restriction\sigma$ defined below in Figure 7.

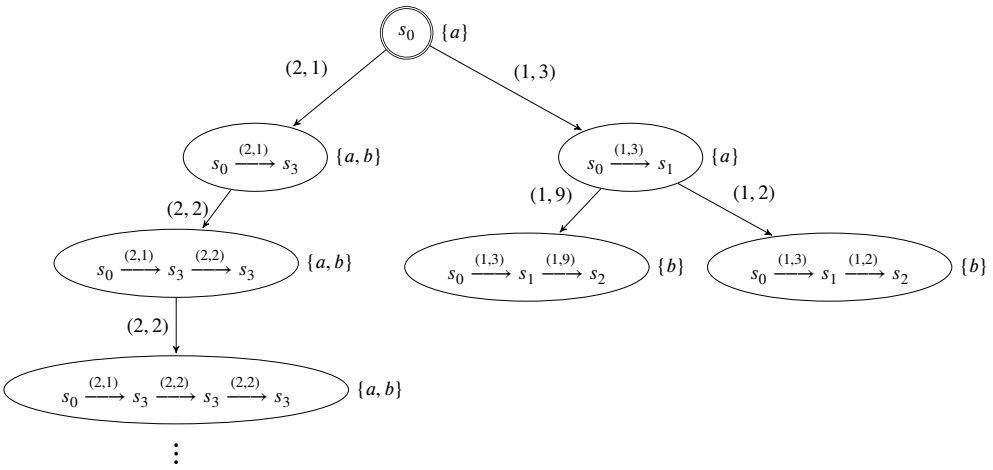


Figure 7: $\mathcal{G}\restriction\sigma$ the resulting n-WKS from restricting $\mathcal{G}$ under σ.

Now we verify that $\mathcal{G}\restriction\sigma, s_0 \models_{0^n} A\ (a)\ U\ ((b \wedge \#2 \geq 8) \vee (b \wedge \#1 \leq 2))$. In $\mathcal{G}\restriction\sigma$ there are three different maximal runs from s_0. Two of these are finite runs and it should be obvious to see that they both satisfy φ. The last run is infinite, however it should be clear by the definition of σ and the illustration of $\mathcal{G}\restriction\sigma$ presented in Figure 7, that we simply repeat the loop $s_3 \xrightarrow{(2,2)} s_3$, thus we have that the run will eventually satisfy φ.

In Definition 4.3 we have defined what we refer to as a *Full Memory Strategy (FM strategy)* where each move in the game graph is recorded in memory, and future decisions

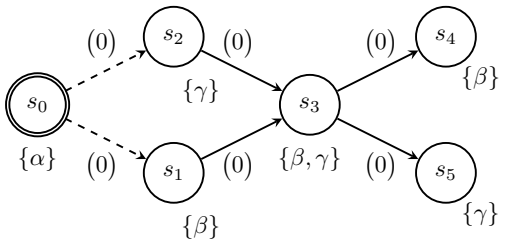
are based thereon. We now introduce a second type of strategy, the *Single-State Cost Strategy (SSC strategy)*, where only the current state is known, along with the cost of the run.

Definition 4.6 (SSC strategy)
A strategy σ is a *Single-State Cost Strategy (SSC strategy)* if for all $\rho, \rho' \in \Pi^{fin}_{K_\mathcal{G}}$ we have $\sigma(\rho) = \sigma(\rho')$, whenever $\textsc{Last}(\rho) = \textsc{Last}(\rho)$ and $cost(\rho) = cost(\rho')$.

We find that if there is a winning strategy for a game with a cb-WCTL winning condition, that strategy cannot always be expressed as an SSC strategy.

Proposition 3
There is a game $(\mathcal{G}, \varphi)$, where φ is a cb-WCTL, with a winning FM strategy, but no winning SSC strategy.

Proof Let $(\mathcal{G}, \varphi)$ be a game, where $\mathcal{G}$ is the 1-WGG illustrated in Figure 8a and φ is the cb-WCTL formula illustrated in Figure 8b. Now, there exists a winning FM strategy σ_1 for



(a) 1-WGG $\mathcal{G}$.

$$\varphi = AG \begin{pmatrix} (\alpha) \vee \\ AG(\beta) \vee \\ AG(\gamma) \end{pmatrix}$$

(b) Formula φ.

Figure 8: Game with a winning FM strategy, but no winning SSC strategy.

$(\mathcal{G}, \varphi)$ defined as:

$$\sigma_1(s_0) = \textsc{nil}$$
$$\sigma_1(s_0 \xrightarrow{(0)} s_1) = s_1 \xrightarrow{(0)} s_3 \qquad \sigma_1(s_0 \xrightarrow{(0)} s_2) = s_2 \xrightarrow{(0)} s_3$$
$$\sigma_1(s_0 \xrightarrow{(0)} s_1 \xrightarrow{(0)} s_3) = s_3 \xrightarrow{(0)} s_4 \qquad \sigma_1(s_0 \xrightarrow{(0)} s_2 \xrightarrow{(0)} s_3) = s_3 \xrightarrow{(0)} s_5$$

However, when we try to define a winning SSC strategy we notice, that for any run ending in s_3, a SSC strategy have to make the same choice as the cost of any run will always be 0. Thus we have that either,

- $\sigma(s_0 \xrightarrow{(0)} s_1 \xrightarrow{(0)} s_3) = \sigma(s_0 \xrightarrow{(0)} s_2 \xrightarrow{(0)} s_3) = s_3 \xrightarrow{(0)} s_4$. This choice is illustrated in Figure 9 and result in an n-WKS with leafs that only satisfy β.
- $\sigma(s_0 \xrightarrow{(0)} s_1 \xrightarrow{(0)} s_3) = \sigma(s_0 \xrightarrow{(0)} s_2 \xrightarrow{(0)} s_3) = s_3 \xrightarrow{(0)} s_5$. This choice results in a similar n-WKS where the leafs only satisfy γ.

Obviously neither strategy is a winning strategy. Thus there exist no winning SSC strategy for $(\mathcal{G}, \varphi)$. □

We have now introduced the notion of a game and a winning strategy and we are ready to define the synthesis problem. The synthesis problem can be informally defined

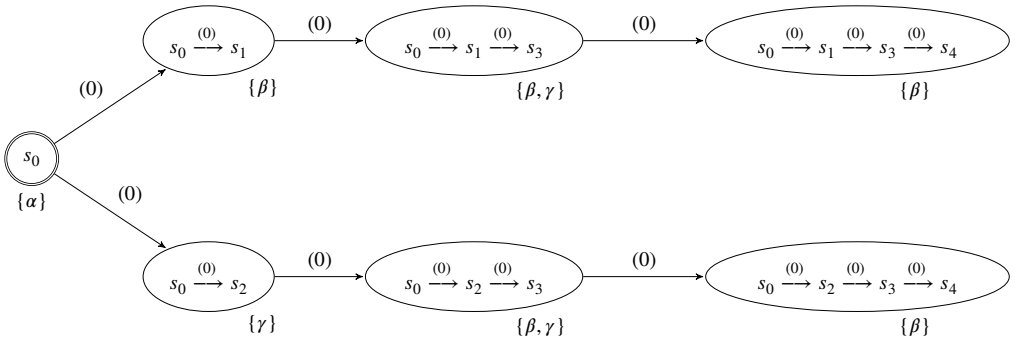


Figure 9: $\mathcal{G}\restriction\sigma$ where σ is a loosing SSC strategy for the game $(\mathcal{G}, \varphi)$.

as: *Determining if there exists a winning strategy for the controller in an n-WG*. We formally define it as:

Definition 4.7 (Synthesis Problem)
Given a game $(\mathcal{G}, \varphi)$ where $\mathcal{G} = (S, s_0, \mathcal{AP}, L, T_c, T_u)$ is an n-WGG and φ is a cb-WCTL formula. The *synthesis problem* is to decide if there is a strategy σ s.t. $\mathcal{G}\restriction\sigma, s_0 \vDash_{0^n} \varphi$.

In the following section we look at the synthesis problem for a game defined with a sub-logics of cb-WCTL describing reachability objectives.

5 Synthesis for Reachability Games

In this Section we introduce the notion of a reachability objective, define the corresponding synthesis problem and provide an algorithmic solution for this problem. The reachability objective is expressed using a sub-logic of cb-WCTL called Reachability WCTL with Upper- and Lower-bounds (ReachWCTL^u_l). Let $K = (S, s_0, \mathcal{AP}, L, T)$ be an n-WKS. We then define the ReachWCTL^u_l syntax, over K, as follows.

$$\varphi := AF\psi$$
$$\psi := a \mid \psi_1 \vee \psi_2 \mid \psi_1 \wedge \psi_2 \mid \#i \bowtie c$$

where $a \in \mathcal{AP}$, $\bowtie \in \{\leq, \geq\}$, $c \in \mathbb{N}_0$, and $1 \leq i \leq n$ is a component index in a vector. A ReachWCTL^u_l formula $AF\psi$ is a ReachWCTL^u formula if all $(\#i \bowtie c) \in Sub(AF\psi)$, are of the form $\#i \leq c$.

We define a *ReachWCTL^u_l game* as a game $(\mathcal{G}, \varphi)$ where $\mathcal{G}$ is n-WGG and φ is a ReachWCTL^u_l formula. We define the corresponding synthesis problem as:

Definition 5.1 (Synthesis Problem for ReachWCTL^u_l Games)
Given a ReachWCTL^u_l game $(\mathcal{G}, \varphi)$ where $\mathcal{G} = (S, s_0, \mathcal{AP}, L, T_c, T_u)$, the *synthesis problem for ReachWCTL^u_l games* is to decide if there is a strategy σ s.t. $\mathcal{G}\restriction\sigma, s_0 \vDash_{0^n} \varphi$.

Similarly, we refer to the synthesis problem for a ReachWCTL^u game as the *synthesis problem for ReachWCTL^u games*.

5.1 Solving the Synthesis Problem for ReachWCTL^u Games

We begin by solving the synthesis problem for ReachWCTL^u games. This will introduce the method for a simpler problem and provide an understanding of the underlying intuition behind the algorithm. The idea behind the method is to first identify all states in the game graph where the objective is trivially satisfied; we refer to these states as *final states*. Then by backwards traversal of the graph we identify all the states which can reach a final state within the cost constraints (without breaching the upper-bound).

We cover the algorithm using the example below throughout Section 5.1 and Section 5.2.

Example 5.1 (Self Driving Car)
Consider a self driving car that needs to find a route from the current location to its destination within its fuel limits and an anti congestion system which may alter the route of the car to avoid traffic congestion. An instance of this problem is shown in the game graph in Figure 10.

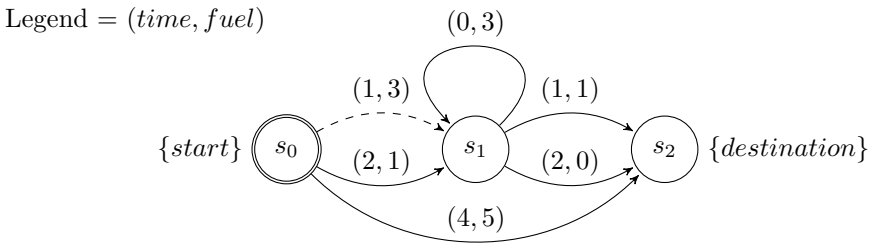


Figure 10: The game graph $\mathcal{G}$ modeling a self driving car and an anti congestion system. The first component on each weight, is the discrete units of time spent, and the second is the units of fuel consumed.

Let us assume the car must reach the destination before 3 units of time has past using only 3 units of fuel. We can now formally state the cars objective as the following ReachWCTL^u formula:

$$\varphi = AF(\#1 \leq 3 \wedge \#2 \leq 3 \wedge destination)$$

In this game we identify s_2 as a final state, since $s_2 \models destination$. Reaching s_2 with any accumulated cost below or equal to the cost constraints will therefore result in the controller winning the game. We see that φ has two cost constraints, $\#1 \leq 3$, and $\#2 \leq 3$ respectively. We now pair s_2 with all vectors for which the state is satisfies the objective. This would result in the following set:

$$\{(s_2, (i,j)) \mid 0 \leq i \leq 3 \text{ and } 0 \leq j \leq 3\}$$

Now we move backwards in the game graph to check if we can add more states, while still keeping the cost within the bound. Because the cost of the transition from s_1 to s_2 is either $(1,1)$ or $(2,0)$, we know that reaching s_1 with any accumulated cost below or equal to $(3-1, 3-1)$ or $(3-2, 3-0)$ means winning the game. We can now pair s_1 with all vectors which fulfill this condition s.t.

$$\{(s_1, (i,j)) \mid 0 \leq i \leq 3-1 \text{ and } 0 \leq j \leq 3-1\}$$
$$\cup$$
$$\{(s_1, (i,j)) \mid 0 \leq i \leq 3-2 \text{ and } 0 \leq j \leq 3\}$$

We repeat the same process for s_0. First we notice, that we can get to s_2 directly from s_0, however the cost of the transition is too high. This means that our strategy has to go

to s_1 from s_0. Unlike for s_1, we can not know for certain which transition we reach s_1 with. Therefore we instead compute the accumulated costs, from which we can win the game, if either transition is taken. The pairs resulting from this are:

$$\begin{gathered}\{(s_0,(i,j)) \mid 0 \leq i \leq 1-1 \text{ and } 0 \leq j \leq 3-3\} \\ \cap \\ \{(s_0,(i,j)) \mid 0 \leq i \leq 2-2 \text{ and } 0 \leq j \leq 2-1\}\end{gathered}$$

Thus we have that the only pair of state and cost, from which we can win from s_0, is $(s_0,(0,0))$.

To keep track of the accumulated cost, as we move backwards in the game graph, we pair a state with a vector $\overline{w} \in \mathbb{N}_0^n$. A pair consisting of a state and a vector is defined as a *configuration*.

Definition 5.2 (Configuration)
Let $\mathcal{G} = (S, s_0, \mathcal{AP}, L, T_c, T_u)$ be a n-WGG. A configuration of $\mathcal{G}$ is a pair $(s, \overline{w})$ where $s \in S$ and $\overline{w} \in \mathbb{N}_0^n$. We denote the set of all configurations as $\mathcal{C}$.

When we have a configuration $(s, \overline{w}) \in \mathcal{C}$ s.t. s can reach a final state with a cost of $\overline{w}$, we say that it is a *winning* configuration. Formally,

Definition 5.3 (Winning configuration)
Let $(\mathcal{G}, \varphi)$ be an ReachWCTL^u game. A configuration $(s, \overline{w})$ is *winning* if there exist a strategy σ s.t. $\mathcal{G}\restriction\sigma, s \models_{\overline{w}} \varphi$.

Let $(\mathcal{G}, \varphi)$ be a ReachWCTL^u game where $\mathcal{G} = (S, s_0, \mathcal{AP}, L, T_c, T_u)$ is an n-WGG and $\varphi = AF\psi$. We can without loss of generality assume that all components have a bound. If they do not, we simply disregard that component in the computation. First we compute the set F_0, which is the set of final configurations.

$$F_0 = \{(s, \overline{w}) \in \mathcal{C} \mid s \models_{\overline{w}} \psi\}$$

Notice, that there are finitely many configurations below the upper bound. The set F_i is then defined inductively for all $i \in \mathbb{N}$.

$$F_i = F_{i-1} \cup \left\{ (s, \overline{w}) \;\middle|\; \begin{array}{l} \text{whenever } s \xrightarrow{\overline{c}} s' \text{ then } (s', \overline{w} + \overline{c}) \in F_{i-1} \text{ and} \\ \text{if } s \rightarrow \text{ then } s \xrightarrow{\overline{c}} s' \text{ s.t. } (s', \overline{w} + \overline{c}) \in F_{i-1} \text{ and} \\ \text{there exists } (s, \overline{c}, s') \in T_c \cup T_u \end{array} \right\}$$

Eventually the sets of winning configurations stabilize s.t. $F_0 \subseteq F_1 \subseteq \ldots F_i = F_{i+1}$ and we denote this last set F_{final}.

Lemma 5
For any finite ReachWCTL^u game $(\mathcal{G}, \varphi)$ there is an $i \in \mathbb{N}_0$ s.t. $F_i = F_{i+1}$.

Proof For any game $(\mathcal{G}, \varphi)$, where $\mathcal{G} = (S, s_0, \mathcal{AP}, L, T_c, T_u)$ is a finite n-WGG, we have that S is also finite. Also observe that the set of possible vectors W each state can be paired with is finite, as each component is either below some bound $c \in \mathbb{N}_0$ or simply disregarded. As both S and W are final sets then so is F_{final}, and thus after repeated application of F_i, eventually $F_i = F_{i+1}$. □

For the set of winning configurations F_{final}, we have that if $(s_0, 0^n) \in F_{final}$ then there exists a winning strategy for $(\mathcal{G}, \varphi)$. We define this strategy one configuration at a time. We define the helping function $Rank : \mathcal{C} \to \mathbb{N}_0$ which for some configuration $(s, \overline{w})$ returns the index of the earliest iteration of F_i where $(s, \overline{w}) \in F_i$.

$$Rank(s, \overline{w}) = min(\{i \mid (s, \overline{w}) \in F_i\})$$

We then map each configuration $(s, \overline{w})$ to a controllable transition $(s, \overline{c}, s') \in T_c$ or NIL if $s \not\to$. Formally we define the set of possible transitions,

$$Possible(s, \overline{w}) = \{s \xrightarrow{\overline{c}} s' \in T_c \mid (s', \overline{w}') \in F_i \text{ s.t. } \overline{w} \leq \overline{w}' - \overline{c} \text{ and } i < Rank(s, \overline{w})\}$$

Now we define that for all runs $\rho, \rho' \in \Pi^{fin}_{K_\mathcal{G}}$ s.t. $\textsc{Last}(\rho) = \textsc{Last}(\rho')$ and $\textsc{Cost}(\rho) = \textsc{Cost}(\rho')$ then,

$$\sigma(\rho) = \sigma(\rho') = \begin{cases} t \in Possible(\textsc{Last}(\rho), \textsc{Cost}(\rho)) & \text{if } Possible(\textsc{Last}(\rho), \textsc{Cost}(\rho)) \neq \emptyset \\ (\textsc{Last}(\rho), \overline{c}, s') \in T_c & \text{else if } \textsc{Last}(\rho) \to \\ \textsc{nil} & \text{else} \end{cases}.$$

Notice the second case will only apply when the configuration is unreachable under the strategy, and is only included to ensure that the strategy is well defined.

Example 5.2 (Self Driving Car)
Consider again Figure 10 from example 5.1. We ascertain that $\psi = (\#1 \leq 3 \wedge \#2 \leq 3 \wedge destination)$, thus $F_0 = \{(s_2, (i, j)) \mid 0 \leq i \leq 3 \text{ and } 0 \leq j \leq 3\}$. We then compute F_i for any $i \in \mathbb{N}$ until $F_i = F_{i+1}$.

$$\begin{aligned} F_1 &= F_0 \cup \{(s_1, (i, j)) \mid 0 \leq i \leq 2 \text{ and } 0 \leq j \leq 2\} \\ &\quad \cup \{(s_1, (i, j)) \mid 0 \leq i \leq 1 \text{ and } 0 \leq j \leq 3\} \\ F_2 &= F_1 \cup \{(s_0, (i, 0)) \mid 0 \leq i \leq 1\} \\ F_3 &= F_2 \cup \emptyset \end{aligned}$$

Thus we have that $F_2 = F_3 = F_{final}$ and from F_{final} we can then extract the SSC strategy σ:

$$\sigma(s_0) = s_0 \xrightarrow{(2,1)} s_1 \qquad \sigma(s_0 \xrightarrow{(1,3)} s_1) = s_1 \xrightarrow{(2,0)} s_2 \qquad \sigma(s_0 \xrightarrow{(2,1)} s_1) = s_1 \xrightarrow{(1,1)} s_2$$

Figure 11 shows the 2-WKS resulting from restricting the game graph $\mathcal{G}$ under the strategy σ.

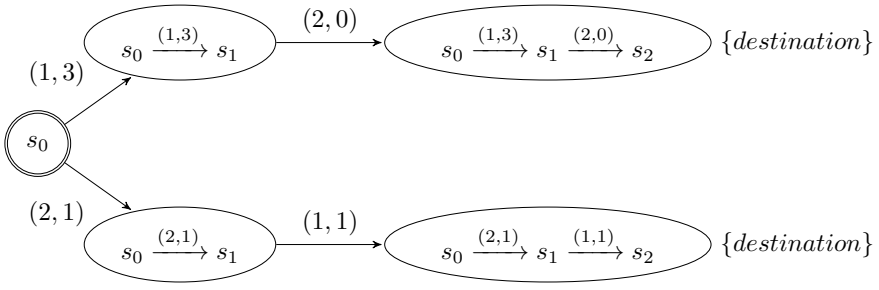


Figure 11: The strategy restricted 2-WKS $\mathcal{G}\restriction\sigma$

Based on the illustration in Figure 11 it should be trivial to confirm that $\mathcal{G}\restriction\sigma, s_0 \vDash_{0^n} AF\psi$ as all possible maximal paths reach a state with the label *destination* within the cost $(3, 3)$.

Lemma 6
If a configuration $(s, \overline{w}) \in F_i$, for some $i \in \mathbb{N}_0$, then $(s, \overline{w})$ is winning.

Proof Proof by induction on i. Let $(\mathcal{G}, \varphi)$ be a ReachWCTL^u game, where $\mathcal{G} = (S, s_0, \mathcal{AP}, L, T_c, T_u)$ is an n-WGG and $\varphi = AF\psi$.

Basis: Follows trivially from the definition of F_0.

Induction step: Assume all configurations in F_i are winning, for some arbitrary i. We will now show that any configuration in F_{i+1} is also winning. Let $(s, \overline{w}) \in F_{i+1}$ be such a configuration. There are two cases:

1. Either $(s, \overline{w}) \in F_i$ and by the induction hypothesis we get that $(s, \overline{w})$ is a winning configuration.
2. Or $(s, \overline{w}) \in F_{i+1} \setminus F_i$ and let σ be a strategy constructed from *Possible*.
   - if $\sigma(s) = \text{NIL}$ then by construction of σ we have that there are no outgoing controllable transitions. By definition of F_i we have that there is some $s \dashrightarrow$ and that whenever $s \overset{\overline{c}}{\dashrightarrow} s'$ there is some $(s', \overline{w}') \in F_i$ s.t. $\overline{w} \leq \overline{w}' - \overline{c}$ and by induction hypothesis $(s', \overline{w}')$ is a winning configuration. Thus σ is a winning strategy and hence $(s, \overline{w})$ is a winning configuration.
   - if $\sigma(s) = s \xrightarrow{\overline{c}} s'$ then by construction of σ we have that $(s', \overline{w}') \in F_i$ s.t. $\overline{w} \leq \overline{w}' - \overline{c}$ and by induction hypothesis we have that $(s', \overline{w}')$ is a winning configuration. By definition of F_i we have that whenever $s \overset{\overline{c}}{\dashrightarrow} s'$ there is some $(s', \overline{w}') \in F_i$ s.t. $\overline{w} \leq \overline{w}' - \overline{c}$ and by induction hypothesis $(s', \overline{w}')$ is a winning configuration. Thus σ is a winning strategy and hence $(s, \overline{w})$ is a winning configuration.

Thus any configuration belonging to some F_i is a winning configuration. □

To prove that any winning configuration $(s, \overline{w}) \in F_i$ we define a depth function over an n-WKS. Given the n-WKS K, we recall that by the semantics of a ReachWCTL^u formula $\varphi = AF\psi$ then if $K, s \vDash_{\overline{w}} \varphi$ we have that for all $(\rho = s_1 \xrightarrow{\overline{c_1}} s_2 \xrightarrow{\overline{c_2}} \dots \xrightarrow{\overline{c_{n-1}}} s_n) \in \Pi_K^{Max}$, where $s = s_1$ then there is some position $i \geq 1$ s.t. $\rho(i) \vDash_{\text{Cost}_i(\rho)} \psi$.

Definition 5.4 (Depth function)
Let $K = (S, s_0, \mathcal{AP}, L, T)$ be an n-WKS and $\varphi = AF\psi$ be a ReachWCTL^u formula. We define a depth function $D : S \times \mathbb{N}_0^n \to \mathbb{N}_\infty$ where

$$D_{(K,\psi)}(s, \overline{w}) = max\{n \mid (\rho = s \to^n s') \text{ and } s' \vDash_{\text{Cost}(\rho)+\overline{w}} \psi\}$$

and $\max(\emptyset) = \infty$.

Finally, recall that given the n-WKS $\mathcal{G}\restriction\sigma = (S', s_0, \mathcal{AP}, L', T_c\restriction\sigma \cup T_u')$ then we have that a state in S' is a finite run $\rho \in \Pi_{K_\mathcal{G}}^{fin}$.

Lemma 7
If a configuration $(s, \overline{w})$ is winning then $(s, \overline{w}) \in F_i$ for some $i \in \mathbb{N}_0$.

Proof Let $(\mathcal{G}, \varphi)$ be a ReachWCTL^u game where $\varphi = AF\psi$. Let $(s, \overline{w})$ be a winning configuration, then there is a strategy σ s.t. $\mathcal{G}\restriction\sigma, s \vDash_{\overline{w}} \varphi$, and then $D_{(\mathcal{G}\restriction\sigma,\varphi)}(s, \overline{w}) = i \in \mathbb{N}_0$.

We prove that if $(s, \overline{w})$ is winning then $(s, \overline{w}) \in F_i$ for some $i \in \mathbb{N}_0$ by showing that if there is a strategy σ where $\mathcal{G}\restriction\sigma, s \vDash_{\overline{w}} \varphi$ s.t. $D_{(\mathcal{G}\restriction\sigma,\varphi)}(s, \overline{w}) = i$ then $(s, \overline{w}) \in F_i$. Proof by induction on i

Basis: If $i = 0$ then by the definition of D, we know that $\mathcal{G}\restriction\sigma, s \vDash_{\overline{w}} \psi$ and by the definition of F_0, $(s, \overline{w})$ must belong to F_0.

Induction step: Assume for all winning configurations $(s', \overline{w}')$ where there is a strategy σ' s.t. $\mathcal{G}\restriction\sigma', s' \vDash_{\overline{w}'} \varphi$ and $D_{(\mathcal{G}\restriction\sigma',\varphi)}(s', \overline{w}') \leq i-1$ then $(s', \overline{w}') \in F_{i-1}$.

As $(s, \overline{w})$ is winning there exists some strategy σ s.t. $\mathcal{G}\restriction\sigma, s \vDash_{\overline{w}} \varphi$. Now, let $\mathcal{G}\restriction\sigma = (S', s_0, \mathcal{AP}, L', T_c\restriction\sigma \cup T'_u)$ and $T = T_c\restriction\sigma \cup T_u$. Clearly, if $s \xrightarrow{\bar{c}} (s \xrightarrow{\bar{c}} s') \in T$ then $\mathcal{G}\restriction\sigma, (s \xrightarrow{\bar{c}} s') \vDash_{\overline{w}+\bar{c}} \varphi$, and for $(s', \overline{w} + \bar{c})$ there is a strategy σ' defined, for any $(\rho = s_1 \xrightarrow{\bar{c}_1} s_2 \xrightarrow{\bar{c}_2} \dots s_n) \in \Pi^{fin}_{K_\mathcal{G}}$, s.t:

$$\sigma'(\rho) = \begin{cases} \sigma((s \xrightarrow{\bar{c}} s_1) \circ \rho) & \text{if } s_1 = s' \text{ and } s \xrightarrow{\bar{c}} (s \xrightarrow{\bar{c}} s') \in T \\ \sigma(\rho) & \text{otherwise} \end{cases}$$

Clearly, $\mathcal{G}\restriction\sigma, (s \xrightarrow{\bar{c}} s') \vDash_{\overline{w}+\bar{c}} \varphi$ iff. $\mathcal{G}\restriction\sigma', s' \vDash_{\overline{w}+\bar{c}} \varphi$. By the definition of σ' we have that $D_{(\mathcal{G},\sigma)}((s \xrightarrow{\bar{c}} s'), \overline{w} + c) = D_{(\mathcal{G},\sigma')}(s', \overline{w} + c) \leq i - 1$. Then by the induction hypothesis we have that $(s', \overline{w} + \bar{c}) \in F_{i-1}$ and then by the definition F_i we know that $(s, \overline{w}) \in F_i$.

Thus if a configuration $(s, \overline{w})$ is winning then there is a strategy σ where $\mathcal{G}\restriction\sigma, s \vDash_{\overline{w}} \varphi$ s.t. $D_{(\mathcal{G}\restriction\sigma,\varphi)}(s, \overline{w}) = i$ then $(s, \overline{w}) \in F_i$. □

Theorem 8 (Complexity of the synthesis problem for ReachWCTL^u games)
There is an algorithm that given a ReachWCTL^u game computes a winning SSC strategy in exponential time.

Proof Let $(\mathcal{G}, \varphi)$ be a ReachWCTL$_l^u$ game where $\mathcal{G} = (S, s_0, \mathcal{AP}, L, T_c, T_u)$ in an n-WGG and $\varphi = AF\psi$. Then by Lemmas 6 and 7 we get that F_i computes a winning strategy, and by the strategy construction we get an SSC strategy, thus such an algorithm exist. Let $|T| = |T_u| + |T_c|$ and let $\overline{b}$ be the boundary vector derived from φ, then $k = \prod_{i=1}^{n} \max(\overline{b}[i], 1)$. Remember that for any $1 \leq i \leq n$ then $-1 \leq \overline{b}[i]$, thus the max function ensure that $k \in \mathbb{N}$. We will now argue for the complexity of computing F_{final} and extracting the strategy.

Computing $\overline{b}$ can be done in linear time in the size of the formula. It should be obvious that for any $(s, \overline{w}) \in F_0$ then $0^n \leq \overline{w} \leq \overline{b}$, hence there can be at most $|S| * k$ configurations. From this, it trivially follows that we have at most $|S| * k$ iterations over F. Assuming each iteration of F takes $O(|T| * |S| * k)$ time and assuming we can extract the strategy in the time it takes to compute an iteration. We then have a time complexity of $O(|T| * |S|^2 * k^2)$. As k is the product of the values of the dimensions of $\overline{b}$, we have that the algorithm run in pseudo-polynomial time in the size of k. □

5.2 Solving the Sythesis Problem for ReachWCTL$_l^u$ Games

We now present an algorithm for solving the synthesis problem for ReachWCTL$_l^u$ games. The algorithm is an extension of the method presented in Section 5.1 for exploring the state space of an n-weighted game. First, notice that the addition of lower-bounds in the logic, adds the possibility of an infinite set of vectors that satisfy the cost constraints of a formula. This will in turn nullify the termination criteria for the algorithm described in Section 5.1.

Example 5.3 (Self Driving Car)
Let us consider the $2-WGG$ $\mathcal{G}$ from Figure 10 in Example 5.1, but now let the game be $(\mathcal{G}, \varphi)$ where $\varphi = AF(\#1 \leq 3 \wedge \#2 \geq 5 \wedge destination)$, so that the game has both upper-

and lower-bounds. Now computing F_0 returns an infinite set of configurations, because we add all vectors where the 2nd component is above 5.

$$F_0 = \{(s_2, (i,j)) \mid 0 \leq i \leq 3 \text{ and } 5 \leq j\},$$

Computing F_i will also return infinite sets, since we now add all vectors where the value of the 2nd component becomes gradually smaller.

$$\begin{aligned}
F_1 &= F_0 \cup \{(s_1, (i,j)) \mid 0 \leq i \leq 2 \text{ and } 4 \leq j\} \\
&\quad \cup \{(s_1, (i,j)) \mid 0 \leq i \leq 1 \text{ and } 5 \leq j\} \\
F_2 &= F_1 \cup \{(s_1, (i,j)) \mid 0 \leq i \leq 2 \text{ and } 1 \leq j\} \\
&\quad \cup \{(s_1, (i,j)) \mid 0 \leq i \leq 1 \text{ and } 2 \leq j\} \\
&\quad \cup \{(s_0, (0,j)) \mid 1 \leq j\} \\
F_3 &= F_2 \cup \{(s_1, (i,j)) \mid 0 \leq i \leq 2 \text{ and } 0 \leq j\} \\
&\quad \cup \{(s_1, (i,j)) \mid 0 \leq i \leq 1 \text{ and } 0 \leq j\} \\
&\quad \cup \{(s_0, (0,j)) \mid 0 \leq j\} \\
F_4 &= F_3 \cup \emptyset
\end{aligned}$$

We have that $F_3 = F_4 = F_{final}$. Note, that while upper-bound games may be won by the presence of a configuration associated with the initial state, lower-bound games can only be won, if there is a winning strategy from the initial state, which does not require the game to start with some cost above 0^n. For example, if we consider the F_{final} we have just computed, there is a winning strategy because the configuration $(s_0, (0,0)) \in F_{final}$.

These infinite sets of vectors can be represented by an interval defined with some lower- and upper-bound based on the cost constraints. As an example, notice that it is possible to define the infinite set F_0 from example 5.3 in the following manner: $F_0 = \{(s_2, \overline{w}) \mid \overline{w} \in [(0,5); (3,\infty)]\}$ where $(0,5) \in \mathbb{N}_0^n$ is a vector defining the lower-bound constraints and $(3,\infty) \in \mathbb{N}_\infty^n$ is a vector defining the upper-bound constraints. We observe that such an interval can be finitely represented by the two vectors defining the end points of the interval, in this example the vectors $(0,5)$ and $(3,\infty)$. We call a tuple of these end points a *zone*, formally:

Definition 5.5 (Zone)
We define the tuple of vectors $(\bar{l}; \overline{u}) \in \mathbb{N}_0^n \times \mathbb{N}_\infty^n$ as a *zone* and define the semantics of a zone as $[\![(\bar{l}; \overline{u})]\!] = \{\overline{w} \mid \bar{l} \leq \overline{w} \leq \overline{u}\}$. We denote the set of all zones as $\mathcal{Z}$.

Based on this representation of vector sets, we define the notion of a symbolic configuration consisting of a state and a zone, instead of a state and a single vector.

Definition 5.6 (Symbolic configuration)
Let $\mathcal{G} = (S, s_0, \mathcal{AP}, L, T_c, T_u)$ be a n-WGG. A symbolic configuration of $\mathcal{G}$ is a pair (s, z) where $s \in S$ and $z \in \mathcal{Z}$. We denote the set of all symbolic configurations as ζ.

Semantically we define the meaning of symbolic configuration as follows:

$$[\![(s,z)]\!] = \{(s, \overline{w}) \mid \overline{w} \in [\![z]\!]\}$$

Notice that there exists symbolic configurations (s, z) where for all $(s, \overline{w}) \in [\![(s, z)]\!]$ it holds that $(s, \overline{w})$ is a winning configuration. We denote these as *winning symbolic configurations*.

Lastly, given a set of symbolic configurations $C \subseteq \zeta$ we have that:

$$[\![C]\!] = \{(s, \overline{w}) \mid (s, z) \in C, \overline{w} \in [\![z]\!]\}.$$

5.2.1 Zones

Now that we have a finite way of representing infinite sets of vectors, we define the operations necessary to adapt the algorithm for ReachWCTL^u games into an algorithm for ReachWCTL$_l^u$ games. We define these as syntactical operations on the zones and show that they are semantically equivalent to operations on sets of vectors.

First we define the notion of a zone being included in another zone.

Definition 5.7 (Symbolic Inclusion of zones)
Let $(\bar{l}; \overline{u}), (\bar{l}'; \overline{u}') \in \mathcal{Z}$ be zones. We define the symbolic inclusion of two zones as the operator $\sqsubseteq$ s.t:

$$(\bar{l}; \overline{u}) \sqsubseteq (\bar{l}'; \overline{u}') \text{ whenever } \bar{l}' \leq \bar{l} \leq \overline{u} \leq \overline{u}'.$$

For a zone z to be included in another zone z' it must hold for all vectors $\overline{w} \in [\![z]\!]$ that $\overline{w} \in [\![z']\!]$.

Lemma 9
Let $z, z' \in \mathcal{Z}$ be zones, then we have that $z \sqsubseteq z'$ iff. $[\![z]\!] \subseteq [\![z']\!]$.

Proof Given $z = (\bar{l}; \overline{u})$ then $[\![(\bar{l}; \overline{u})]\!] = \{\overline{w} \mid \bar{l} \leq \overline{w} \leq \overline{u}\}$ and given $z' = (\bar{l}'; \overline{u}')$ then $[\![(\bar{l}'; \overline{u}')]\!] = \{\overline{w} \mid \bar{l}' \leq \overline{w} \leq \overline{u}'\}$.

- $\Rightarrow$ Assume $z \sqsubseteq z'$ then $\overline{l'} \leq \bar{l} \leq \overline{u} \leq \overline{u'}$. For all $\overline{w}$ where $\bar{l} \leq \overline{w} \leq \overline{u}$ then as $\bar{l}' \leq \bar{l} \leq \overline{u} \leq \overline{u}'$ we have that $\bar{l}' \leq \overline{w} \leq \overline{u}'$. Hence if $z \sqsubseteq z'$ then $[\![z]\!] \subseteq [\![z']\!]$.
- $\Leftarrow$ Assume $[\![z]\!] \subseteq [\![z']\!]$ then for all $\bar{l} \leq \overline{w} \leq \overline{u}$ then $\overline{l'} \leq \overline{w} \leq \overline{u'}$. Thus $\overline{l'} \leq \bar{l} \leq \overline{u} \leq \overline{u'}$. Hence if $[\![z]\!] \subseteq [\![z']\!]$ then $z \sqsubseteq z'$. □

Next we define subtraction of a vector from a zone.

Definition 5.8 (Subtracting vector from zone)
Let $(\bar{l}; \overline{u}) \in \mathcal{Z}$ be a zone. We define the subtraction operator $\div$ for subtracting a vector $\overline{c} \in \mathbb{N}_0^n$ from a zone component-wise s.t. for each $1 \leq i \leq n$

$$((\bar{l}; \overline{u}) \div \overline{c})[i] = \begin{cases} (\bar{l}[i] - \overline{c}[i]; \overline{u}[i] - \overline{c}[i]) & \text{if } 0 \leq \bar{l}[i] - \overline{c}[i] \\ (0; \overline{u}[i] - \overline{c}[i]) & \text{otherwise} \end{cases}.$$

If for any $1 \leq i \leq n$, it holds that $(\overline{u}[i] - \overline{c}[i]) < 0$ then $(\bar{l}; \overline{u}) \div \overline{c} = \bot$

We want to ensure that when we subtract a constant $\overline{c}$ from a zone z then the new zone $z \div \overline{c}$ represents the set of vectors $\overline{w} \in \mathbb{N}_0^n$ where $\overline{w} + \overline{c} \in [\![z]\!]$. Given $z = (\bar{l}; \overline{u})$ then if subtracting the constant $\overline{c}$ pushes the upper-bound $\overline{u}$ below zero we return $\bot$ instead with the semantics defined as $[\![\bot]\!] = \emptyset$.

Lemma 10
Let $z \in \mathcal{Z}$ be a zone and $\overline{c} \in \mathbb{N}_0^n$ be a vector, then $[\![z \div \overline{c}]\!] = \{\overline{w} \in \mathbb{N}_0^n \mid \overline{w} + \overline{c} \in [\![z]\!]\}$.

Proof Let $z = (\overline{l}; \overline{u})$ where $\overline{l} \leq \overline{u}$, $[\![(\overline{l}; \overline{u})]\!] = \{\overline{w} \mid \overline{l} \leq \overline{w} \leq \overline{u}\}$ and $\overline{c} \in \mathbb{N}_0^n$. To prove $[\![z \div \overline{c}]\!] = \{\overline{w} \in \mathbb{N}_0^n \mid \overline{w} + \overline{c} \in [\![z]\!]\}$ we show for any $\overline{w} \in \mathbb{N}_0^n$ that $\overline{w} \in [\![z \div \overline{c}]\!]$ iff. $\overline{w} \in \{\overline{w} \in \mathbb{N}_0^n \mid \overline{w} + \overline{c} \in [\![z]\!]\}$.

$\Rightarrow$ For any $\overline{w} \in [\![(\overline{l}; \overline{u}) \div \overline{c}]\!]$ we want to show that $\overline{w} \in \{\overline{w} \in \mathbb{N}_0^n \mid \overline{w} + \overline{c} \in [\![(\overline{l}; \overline{u})]\!]\}$. Let $(\overline{l}'; \overline{u}') = (\overline{l}; \overline{u}) \div \overline{c}$. Then we have two cases:

- If $(\overline{l}'; \overline{u}') = \bot$ then $[\![(\overline{l}'; \overline{u}')]\!] = \emptyset$. It then follows trivially that all $\overline{w} \in [\![(\overline{l}'; \overline{u}')]\!]$ is also $\overline{w} \in \{\overline{w} \in \mathbb{N}_0^n \mid \overline{w} + \overline{c} \in [\![(\overline{l}; \overline{u})]\!]\}$.
- If $(\overline{l}'; \overline{u}') \neq \bot$ then $[\![(\overline{l}'; \overline{u}')]\!] \neq \emptyset$. Then we have that $\overline{l}' + \overline{c} \in \{\overline{w} \in \mathbb{N}_0^n \mid \overline{w} + \overline{c} \in [\![(\overline{l}; \overline{u})]\!]\}$ and $\overline{u}' + \overline{c} \in \{\overline{w} \in \mathbb{N}_0^n \mid \overline{w} + \overline{c} \in [\![(\overline{l}; \overline{u})]\!]\}$. Hence for all $\overline{w} \in [\![(\overline{l}'; \overline{u}')]\!]$ is also $\overline{w} \in \{\overline{w} \in \mathbb{N}_0^n \mid \overline{w} + \overline{c} \in [\![(\overline{l}; \overline{u})]\!]\}$.

$\Leftarrow$ For any $\overline{w} \in \{\overline{w} \in \mathbb{N}_0^n \mid \overline{w} + \overline{c} \in [\![(\overline{l}; \overline{u})]\!]\}$ we want to show that $\overline{w} \in [\![(\overline{l}; \overline{u}) \div \overline{c}]\!]$. Let $(\overline{l}'; \overline{u}') = (\overline{l}; \overline{u}) \div \overline{c}$. Then we have two cases:

- If there exist an $1 \leq i \leq n$ s.t. $\overline{u}[i] - \overline{c}[i] < 0$, then for all $\overline{w}' \in \mathbb{N}_0^n$ it holds that $\overline{w}' + \overline{c} \not\leq \overline{u}$, and thus $\{\overline{w} \in \mathbb{N}_0^n \mid \overline{w} + \overline{c} \in [\![(\overline{l}; \overline{u})]\!]\} = \emptyset$. We then also have that $(\overline{l'}; \overline{u'}) = \bot$. Hence, any $\overline{w} \in \{\overline{w} \in \mathbb{N}_0^n \mid \overline{w} + \overline{c} \in [\![(\overline{l}; \overline{u})]\!]\}$ is also $\overline{w} \in [\![(\overline{l}'; \overline{u}')]\!]$
- If for all $1 \leq i \leq n$ it holds that $\overline{u}[i] - \overline{c}[i] \geq 0$. Then for any $\overline{w} \in \{\overline{w} \in \mathbb{N}_0^n \mid \overline{w} + \overline{c} \in [\![(\overline{l}; \overline{u})]\!]\}$ we then have that $\overline{w} \leq \overline{u} - \overline{c} = \overline{u}'$, and that $\overline{w} \geq \overline{l}' \geq \overline{l} - \overline{c}$, as either $\overline{l} - \overline{c} = \overline{l}'$ or $\overline{l} - \overline{c} \notin \mathbb{N}_0^n$. Hence, for any $\overline{w} \in \{\overline{w} \in \mathbb{N}_0^n \mid \overline{w} + \overline{c} \in [\![(\overline{l}; \overline{u})]\!]\}$ is also $\overline{w} \in [\![(\overline{l}; \overline{u}) \div \overline{c}]\!]$.

Hence $\overline{w} \in [\![(\overline{l}; \overline{u}) \div \overline{c}]\!]$ iff $\overline{w} \in \{\overline{w} \mid \overline{w} + \overline{c} \in [\![(\overline{l}; \overline{u})]\!]\}$ □

Next we define the symbolic intersection of two zones.

Definition 5.9 (Symbolic Zone Intersection)
Let $(\overline{l}; \overline{u}), (\overline{l}'; \overline{u}') \in \mathcal{Z}$ be zones. We define the symbolic intersection of the two zones as the operator $\sqcap$ for intersecting component-wise s.t. for each $1 \leq i \leq n$

$$((\overline{l}; \overline{u}) \sqcap (\overline{l}'; \overline{u}'))[i] = (max(\overline{l}[i], \overline{l}'[i]); min(\overline{u}[i], \overline{u}'[i]))$$

If for any $1 \leq i \leq n$, it holds that $max(\overline{l}[i], \overline{l}'[i]) > min(\overline{u}[i], \overline{u}'[i])$ then $(\overline{l}; \overline{u}) \sqcap (\overline{l}'; \overline{u}') = \bot$.

We want the symbolic intersection operation to return a zone that semantically represents the set of vectors which are semantically in both zones.

Lemma 11
Let $z, z' \in \mathcal{Z}$ be zones, then $[\![z \sqcap z']\!] = [\![z]\!] \cap [\![z']\!]$.

Proof Given $z = (\overline{l}; \overline{u})$ then $[\![(\overline{l}; \overline{u})]\!] = \{\overline{w} \mid \overline{l} \leq \overline{w} \leq \overline{u}\}$ and given $z' = (\overline{l}'; \overline{u}')$ then $[\![(\overline{l}'; \overline{u}')]\!] = \{\overline{w} \mid \overline{l}' \leq \overline{w} \leq \overline{u}'\}$. Now, there are two directions:

$\Rightarrow$: We will now prove that $[\![z \sqcap z']\!] \subseteq [\![z]\!] \cap [\![z']\!]$. There are two cases:

- If there is some $1 \leq i \leq n$ s.t. $max(\overline{l}[i], \overline{l}'[i]) > min(\overline{u}[i], \overline{u}'[i])$ then $[\![z \sqcap z']\!] = \emptyset$, and obviously $\emptyset \subseteq [\![z]\!] \cap [\![z']\!]$.

- For any $\overline{w} \in [\![z \sqcap z']\!]$ and any $1 \leq i \leq n$ we have that $max(\overline{l}[i], \overline{l}'[i]) \leq \overline{w}[i] \leq min(\overline{u}[i], \overline{u}'[i])$. Then it must hold that $\overline{l}' \leq \overline{w} \leq \overline{u}'$ and $\overline{l} \leq \overline{w} \leq \overline{u}$, and then we have that $\overline{w} \in [\![z]\!] \cap [\![z']\!]$. Hence $[\![z \sqcap z']\!] \subseteq [\![z]\!] \cap [\![z']\!]$.

$\Leftarrow$: We will now prove that $[\![z]\!] \cap [\![z']\!] \subseteq [\![z \sqcap z']\!]$. For any $\overline{w} \in [\![z]\!] \cap [\![z']\!]$ it holds that $\overline{l}' \leq \overline{w} \leq \overline{u}'$ and $\overline{l} \leq \overline{w} \leq \overline{u}$. Then for any $1 \leq i \leq n$ it must hold that $max(\overline{l}[i], \overline{l}'[i]) \leq \overline{w}[i] \leq min(\overline{u}[i], \overline{u}'[i])$ and as such $\overline{w} \in [\![z \sqcap z']\!]$. Hence $[\![z]\!] \cap [\![z']\!] \subseteq [\![z \sqcap z']\!]$.

Hence, as $[\![z \sqcap z']\!] \subseteq [\![z]\!] \cap [\![z']\!]$ and $[\![z]\!] \cap [\![z']\!] \subseteq [\![z \sqcap z']\!]$ then $[\![z \sqcap z']\!] = [\![z]\!] \cap [\![z']\!]$. □

The last operation we define is a trivial extension of symbolic intersection to sets of zones.

Definition 5.10 (Symbolic Intersection of Sets of Zones)
Let $Z, Z' \subseteq \mathcal{Z}$ be sets of zones. We define the symbolic intersection of Z and Z' s.t.

$$Z \sqcap Z' = \{z \sqcap z' \mid z \in Z \text{ and } z' \in Z' \text{ and } z \sqcap z' \neq \bot\}$$

whenever $Z \neq \emptyset$ and $Z' \neq \emptyset$.

This will return the set of vectors which represented in both sets by some zone.

Lemma 12
Let $Z, Z' \subseteq \mathcal{Z}$ be sets of zones, then $[\![Z \sqcap Z']\!] = \{\overline{w} \mid \overline{w} \in [\![z \sqcap z']\!] \text{ where } z \in Z \text{ and } z' \in Z'\}$.

Proof This follows trivially from definition 5.10 and Lemma 11. □

5.2.2 Calculating F_0 with Zones

We now present the method for calculating the initial set of winning symbolic configurations $\mathcal{F}_0$ s.t. $[\![\mathcal{F}_0]\!] = F_0$. Recall that F_0 for ReachWCTL^u games is defined as the set of all final configurations. Intuitively this definition still holds, but now we need to find the set of symbolic configurations which represents the set of all final configurations.

Through a traversal of the objective, we find all upper- and lower-bounds specified. We then combine these into zones depending on the logical operands of the objective and a given state. The algorithm for constructing these zones is given in Algorithm 2. The idea is to start with the largest possible zone, $(0^n; \infty^n)$, and then shrink it, whenever we reach a cost constraint. If the constraints are conjunctive, the zone is shrunk, and if they are disjunctive, two separate zones are returned, each bounded by their respective constraints.

The zones created by Algorithm 2, have two properties that make them a suitable finite representation for the infinite vector sets. First and foremost, the algorithm returns a finite set and secondly it precisely represents the set of trivially winning symbolic configurations of the input state.

Lemma 13
Let $(\mathcal{G}, \varphi)$ be a ReachWCTL$_l^u$ game, where $\mathcal{G} = (S, s_0, \mathcal{AP}, L, T_c, T_u)$ and $\varphi = AF\psi$. Given a state $s \in S$ and a zone $(\overline{l}; \overline{u}) \in \mathcal{Z}$ then GetBasis$(\psi, (\overline{l}; \overline{u}), s)$ will return a finite set of zones.

Proof Proof by structural induction on ψ.

- Let ψ be some $a \in \mathcal{AP}$. Then only one zone is returned, thus the set of zones is finite.

Algorithm 2 Algorithm computing a set of zones Z where $s \vDash_{\overline{w}} \psi$ for any $\overline{w} \in [\![Z]\!]$.

Input: A formula ψ, a zone $(\bar{l}; \overline{u})$ and a state s.
Output: A subset of $\mathcal{Z}$

1: **function** GetBasis($\psi, (\bar{l}; \overline{u}), s$) ▷ Assume $\mathcal{G} = (S, s_0, \mathcal{AP}, L, T_c, T_u)$.
2: **if** ψ is some $a \in \mathcal{AP}$ **then**
3: **if** $s \vDash a$ **then**
4: **return** $\{(\bar{l}; \overline{u})\}$
5: **else return** $\emptyset$
6: **if** $\psi = \#i \leq c$ **then**
7: $\overline{u}[i] \leftarrow min(\overline{u}[i], c)$
8: **if** $\bar{l}[i] > \overline{u}[i]$ **then return** $\emptyset$
9: **return** $\{(\bar{l}; \overline{u})\}$
10: **if** $\psi = \#i \geq c$ **then**
11: $\bar{l}[i] \leftarrow max(\bar{l}[i], c)$
12: **if** $\bar{l}[i] > \overline{u}[i]$ **then return** $\emptyset$
13: **return** $\{(\bar{l}; \overline{u})\}$
14: **if** $\psi = \psi_1 \wedge \psi_2$ **then**
15: **return** GetBasis$(\psi_1, (\bar{l}; \overline{u}), s) \sqcap$ GetBasis$(\psi_2, (\bar{l}; \overline{u}), s)$
16: **if** $\psi = \psi_1 \vee \psi_2$ **then**
17: **return** GetBasis$(\psi_1, (\bar{l}; \overline{u}), s) \cup$ GetBasis$(\psi_2, (\bar{l}; \overline{u}), s)$

- Let $\psi = \#i \leq c$, then one zone is returned, thus the set of zones is finite.
- Let $\psi = \#i \geq c$, then one zone is returned, thus the set of zones is finite.
- Let $\psi = \psi_1 \wedge \psi_2$. Assume GetBasis$(\psi_1, (\bar{l}; \overline{u}), s)$ and GetBasis$(\psi_2, (\bar{l}; \overline{u}), s)$ both return a finite set of zones. The intersection of two finite sets is also a finite set, thus the set of zones is finite.
- Let $\psi = \psi_1 \vee \psi_2$. Assume GetBasis$(\psi_1, (\bar{l}; \overline{u}), s)$ and GetBasis$(\psi_2, (\bar{l}; \overline{u}), s)$ both return a finite set of zones. The union of two finite sets is also a finite set, thus the set of zones is finite. □

Lemma 14
Let $(\mathcal{G}, \varphi)$ be a reachability game where $\mathcal{G} = (S, s_0, \mathcal{AP}, L, T_c, T_u)$ and $\varphi = AF\psi$. For any $s \in S$ and any $(\bar{l}; \overline{u}) \in$ GetBasis$(\psi, (0^n; \infty^n), s)$ it holds that for all $\bar{l} \leq \overline{w} \leq \overline{u}$ that $s \vDash_{\overline{w}} \psi$.

Proof Assume there exist $(\bar{l}; \overline{u}) \in$ GetBasis$(\psi, (0^n; \infty^n), s)$, where $s \in S$. Proof by structural induction on ψ.

- $\boldsymbol{\psi = a \in \mathcal{AP}}$**:** Then GetBasis$(\psi, (0^n; \infty^n), s) = \{(0^n; \infty^n)\}$ and by the semantics it holds that $s \vDash_{\overline{w}} a$ for any $0^n \leq \overline{w} \leq \infty^n$.
- $\boldsymbol{\psi = \#i \geq c}$**:** Then GetBasis$(\psi, (0^n; \infty^n), s) = \{(0^n[i \to c]; \infty^n)\}$ and by the semantics it holds that $s \vDash_{\overline{w}} \#i \geq c$ for any $0^n[i \to c] \leq \overline{w} \leq \infty^n$.
- $\boldsymbol{\psi = \#i \leq c}$**:** Then GetBasis$(\psi, (0^n; \infty^n), s) = \{(0^n; \infty^n[i \to c])\}$ and by the semantics it holds that $s \vDash_{\overline{w}} \#i \leq c$ for any $0^n \leq \overline{w} \leq \infty^n[i \to c]$.
- $\boldsymbol{\psi = \psi_1 \vee \psi_2}$**:** By the induction hypothesis it holds that $s \vDash_{\overline{w}} \psi_1$ for any $\overline{w}$ s.t. $\bar{l}_1 \leq \overline{w} \leq \overline{u}_1$ where $(\bar{l}_1; \overline{u}_1) \in$ GetBasis$(\psi_1, (0^n; \infty^n), s)$. It also holds that $s \vDash_{\overline{w}} \psi_2$ for any $\overline{w}$ s.t. $\bar{l}_2 \leq \overline{w} \leq \overline{u}_2$ where $(\bar{l}_2; \overline{u}_2) \in$ GetBasis$(\psi_2, (0^n; \infty^n), s)$. As GetBasis$(\psi, (0^n; \infty^n), s) =$ GetBasis$(\psi_1, (0^n; \infty^n), s) \cup$ GetBasis$(\psi_2, (0^n; \infty^n), s)$ it follows that $s \vDash_{\overline{w}} \psi$ for any $\overline{w}$ s.t. $\bar{l} \leq \overline{w} \leq \overline{u}$ where $(\bar{l}; \overline{u}) \in$ GetBasis$(\psi, (0^n; \infty^n), s)$.

$\boldsymbol{\psi = \psi_1 \wedge \psi_2}$**:** By the induction hypothesis it holds that $s \vDash_{\overline{w}} \psi_1$ for any $\overline{w}$ s.t. $\overline{l}_1 \leq \overline{w} \leq \overline{u}_1$ where $(\overline{l}_1; \overline{u}_1) \in \textsc{GetBasis}(\psi_1, (0^n; \infty^n), s)$. It also holds that $s \vDash_{\overline{w}} \psi_2$ for any $\overline{w}$ s.t. $\overline{l}_2 \leq \overline{w} \leq \overline{u}_2$ where $(\overline{l}_2; \overline{u}_2) \in \textsc{GetBasis}(\psi_2, (0^n; \infty^n), s)$. As $\textsc{GetBasis}(\psi, (0^n; \infty^n), s) = \textsc{GetBasis}(\psi_1, (0^n; \infty^n), s) \sqcap \textsc{GetBasis}(\psi_2, (0^n; \infty^n), s)$ it then follows from Lemma 12 that $s \vDash_{\overline{w}} \psi$ for any $\overline{w}$ s.t. $\overline{l} \leq \overline{w} \leq \overline{u}$ where $(\overline{l}; \overline{u}) \in \textsc{GetBasis}(\psi, (0^n; \infty^n), s)$.

Hence for any $s \in S$ and any $(\overline{l}; \overline{u}) \in \textsc{GetBasis}(\psi, (0^n; \infty^n), s)$ it holds that for all $\overline{l} \leq \overline{w} \leq \overline{u}$ that $s \vDash_{\overline{w}} \psi$, where $\varphi = AF\psi$ is a ReachWCTL$_l^u$ formula. □

Lemma 15
Let $(\mathcal{G}, \varphi)$ be a reachability game where $\mathcal{G} = (S, s_0, \mathcal{AP}, L, T_c, T_u)$ and $\varphi = AF\psi$. For any $s \in S$ and any $\overline{w} \in \mathbb{N}_\infty^n$ it holds that if $s \vDash_{\overline{w}} \psi$ then there exists $(\overline{l}; \overline{u}) \in \textsc{GetBasis}(\psi, (0^n; \infty^n), s)$ where $\overline{l} \leq \overline{w} \leq \overline{u}$.

Proof Assume $s \vDash_{\overline{w}} \psi$ for some $\overline{w} \in \mathbb{N}_\infty^n$. Proof by structural induction on ψ.

$\boldsymbol{\psi = a \in \mathcal{AP}}$**:** Since $s \vDash_{\overline{w}} a$ for any $\overline{w} \in \mathbb{N}_\infty^n$ and $\textsc{GetBasis}(\psi, (0^n; \infty^n), s) = \{(0^n; \infty^n)\}$, it follows trivially that there exists $(\overline{l}; \overline{u}) \in \textsc{GetBasis}(\psi, (0^n; \infty^n), s)$ where $\overline{l} \leq \overline{w} \leq \overline{u}$.

$\boldsymbol{\psi = \#i \geq c}$**:** We have that $s \vDash_{\overline{w}} \psi$ for any $\overline{w} \in \mathbb{N}_0^n$ where $\overline{w}[i] \geq c$. As $\textsc{GetBasis}(\psi, (0^n; \infty^n), s) = \{(0^n[i \to c]; \infty^n)\}$, there exists $(\overline{l}; \overline{u}) \in \{(0^n[i \to c]; \infty^n)\}$ s.t. $\overline{l} \leq \overline{w} \leq \overline{u}$ for any $\overline{w} \in \mathbb{N}_\infty^n$ where $\overline{w}[i] \geq c$.

$\boldsymbol{\psi = \#i \leq c}$**:** We have that $s \vDash_{\overline{w}} \psi$ for any $\overline{w} \in \mathbb{N}_0^n$ where $\overline{w}[i] \leq c$. As $\textsc{GetBasis}(\psi, (0^n; \infty^n), s) = \{(0^n; \infty^n[i \to c])\}$, there exists $(\overline{l}; \overline{u}) \in \{(0^n; \infty^n[i \to c])\}$ s.t. $\overline{l} \leq \overline{w} \leq \overline{u}$ for any $\overline{w} \in \mathbb{N}_\infty^n$ where $\overline{w}[i] \leq c$.

$\boldsymbol{\psi = \psi_1 \vee \psi_2}$**:** We have that $s \vDash_{\overline{w}} \psi$ iff. $s \vDash_{\overline{w}} \psi_1$ or $s \vDash_{\overline{w}} \psi_2$. By the induction hypothesis, it holds that if $s \vDash_{\overline{w}} \psi_1$ then there exists $(\overline{l}_1; \overline{u}_1) \in \textsc{GetBasis}(\psi_1, (0^n; \infty^n), s)$ s.t. $s \vDash_{\overline{w}} \psi_1$ for any $\overline{l}_1 \leq \overline{w} \leq \overline{u}_1$. It also holds that if $s \vDash_{\overline{w}} \psi_2$ then there exists $(\overline{l}_2; \overline{u}_2) \in \textsc{GetBasis}(\psi_2, (0^n; \infty^n), s)$ s.t. $s \vDash_{\overline{w}} \psi_2$ for any $\overline{l}_2 \leq \overline{w} \leq \overline{u}_2$. Thus exist $(\overline{l}; \overline{u}) \in \textsc{GetBasis}(\psi, (0^n; \infty^n), s) = \textsc{GetBasis}(\psi_1, (0^n; \infty^n), s) \cup \textsc{GetBasis}(\psi_2, (0^n; \infty^n), s)$ where $s \vDash_{\overline{w}} \psi$ for any $\overline{l} \leq \overline{w} \leq \overline{u}$.

$\boldsymbol{\psi = \psi_1 \wedge \psi_2}$**:** We have that $s \vDash_{\overline{w}} \psi$ iff. $s \vDash_{\overline{w}} \psi_1$ and $s \vDash_{\overline{w}} \psi_2$. By the induction hypothesis, there exists $(\overline{l}_1; \overline{u}_1) \in \textsc{GetBasis}(\psi_1, (0^n; \infty^n), s)$ where $\overline{l}_1 \leq \overline{w} \leq \overline{u}_1$ and $(\overline{l}_2; \overline{u}_2) \in \textsc{GetBasis}(\psi_2, (0^n; \infty^n), s)$ where $\overline{l}_2 \leq \overline{w} \leq \overline{u}_2$. Thus exist $(\overline{l}; \overline{u}) \in \textsc{GetBasis}(\psi, (0^n; \infty^n), s) = \textsc{GetBasis}(\psi_1, (0^n; \infty^n), s) \sqcap \textsc{GetBasis}(\psi_2, (0^n; \infty^n), s)$ where $\overline{l}_1 \leq \overline{l} \leq \overline{w} \leq \overline{u} \leq \overline{u}_1$ and $\overline{l}_2 \leq \overline{l} \leq \overline{w} \leq \overline{u} \leq \overline{u}_2$ s.t. $s \vDash_{\overline{w}} \psi$ for any $\overline{l} \leq \overline{w} \leq \overline{u}$.

Hence there exists $(\overline{l}; \overline{u}) \in \textsc{GetBasis}(\psi, (0^n; \infty^n), s)$ s.t. $s \vDash_{\overline{w}} \psi$ for any $\overline{l} \leq \overline{w} \leq \overline{u}$. □

With the properties proven, we can now define $\mathcal{F}_0$ for a ReachWCTL$_l^u$ game $(\mathcal{G}, AF\psi)$, as the set of winning symbolic configurations:

$$\mathcal{F}_0 = \{(s, z) \mid s \in S \text{ and } z \in \textsc{GetBasis}(\psi, (0^n; \infty^n), s)\}$$

By Lemma 2 $\mathcal{F}_0$ is finite, by Lemma 14 and Lemma 15 we have that $[\![\mathcal{F}_0]\!] = F_0$.

5.2.3 Calculating F_i with Zones

With $\mathcal{F}_0$ defined as a finite set of winning symbolic configurations, we are ready to create $\mathcal{F}_i$. Recall that for ReachWCTL^u games F_i computes the set of winning configurations which can reach a configuration in F_{i-1} in a single step. We will now generalize this notion for symbolic configurations.

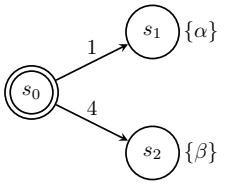
First we notice that we can make a naive reimplementation of F_i based on the zone operations defined in Section 5.2.1. However this would be inefficient as we would generate a lot of redundant information. This approach is illustrated in Example 5.4.

Example 5.4 (Naive modification of F_i)
Recall that $\mathcal{F}_0$ is now defined as a set of winning symbolic configurations based on Algorithm 2. With subtraction for symbolic configurations defined in Definition 5.8 and symbolic inclusion defined in Definition 5.7, we can make a naive definition of $\mathcal{F}_i$ looking like this:

$$\mathcal{F}_i = \mathcal{F}_{i-1} \cup \left\{ (s, z \div \overline{c}) \;\middle|\; \begin{array}{l} \text{whenever } s \overset{\overline{c}}{\dashrightarrow} s' \text{ then } (s', z') \in \mathcal{F}_{i-1}, \text{ where } z \sqsubseteq z' \text{ and} \\ \text{if } s \rightarrow \text{ then } s \overset{\overline{c}}{\rightarrow} s' \text{ s.t. } (s', z') \in \mathcal{F}_{i-1}, \text{ where } z \sqsubseteq z' \text{ and} \\ \text{there exists } (s, \overline{c}, s') \in T_c \cup T_u \text{ and } z \div \overline{c} \neq \bot \end{array} \right\}$$

While symbolic configurations can be used to represent infinitely many configurations in a concise manner, the above definition can, potentially, produce an unnecessary amount symbolic configurations.

Consider the ReachWCTL^u_l game in Figure 12. Notice that we omit the vector notation as $\mathcal{G}$ is a 1-WGG.



(a) $\mathcal{G} = (S, s_0, \mathcal{AP}, L, T_c, T_u)$

$$\varphi = AF \begin{pmatrix} (\{\alpha\} \wedge \#1 \geq 1) \\ \vee \\ (\{\beta\} \wedge \#1 \geq 3) \end{pmatrix}$$

(b) ReachWCTL^u_l formula φ

Figure 12: ReachWCTL^u_l game

By the definition of $\mathcal{F}_0$ we have that,

$$\mathcal{F}_0 = \{(s_1, (1; \infty)), (s_2, (3; \infty))\}.$$

Now, repeated application of $\mathcal{F}_i$ is shown below:

$$\begin{aligned} \mathcal{F}_1 =& \mathcal{F}_0 \cup \{(s_0, z \div 1) \mid z \sqsubseteq (1; \infty)\} \cup \{(s_0, z \div 4) \mid z \sqsubseteq (3; \infty)\} \\ \mathcal{F}_2 =& \mathcal{F}_1 = \mathcal{F}_{final} \end{aligned}$$

$\mathcal{F}_1$ is an infinite set, as there are infinitely many zones $z \in \mathcal{Z}$ s.t. $z \sqsubseteq (1; \infty)$ or $z \sqsubseteq (3; \infty)$. To avoid this we can construct an artificial upper-bound, such that the computation of $\mathcal{F}_i$ will return a finite set. However, this approach would still result in a large amount of irrelevant symbolic configurations being added.

Thus instead of using this naive implementation we present the function $Next : \mathcal{P}(\zeta) \rightarrow \mathcal{P}(\zeta)$. $Next$ takes a set of symbolic configurations $\mathcal{F} \subseteq \zeta$ as input, and returns the set of symbolic configurations, which can reach a symbolic configuration in $\mathcal{F}$ in a single transition. As with F_i the idea is that if $\mathcal{F}$ is a set of winning symbolic configurations, then $Next(\mathcal{F})$ is also a set of winning symbolic configurations. The pseudo-code for $Next$ is presented in Algorithm 3.

Lemma 16
Let $\mathcal{G} = (S, s_0, \mathcal{AP}, L, T_c, T_u)$ be an n-WGG, then given a finite set of symbolic configurations $\mathcal{F} \subseteq \zeta$ as input, $Next(\mathcal{F})$ is a finite set of symbolic configurations.

Algorithm 3 Algorithms for computing $\mathcal{F}_i$

Input: A finite set of symbolic configurations $\mathcal{F}$.
Output: A finite set of symbolic configurations C.

1: **function** $Next(\mathcal{G}, \mathcal{F})$ ▷ Assume $\mathcal{G} = (S, s_0, \mathcal{AP}, L, T_c, T_u)$.
2: $C \leftarrow \emptyset$
3: **for all** $s \in S$ **do**
4: $Z_u \leftarrow UZones(s, T_u, \mathcal{F})$
5: $Z_c \leftarrow \{(0^n; \infty^n)\}$
6: **if** $\exists (s, \overline{c}, s') \in T_c$ **then**
7: $Z_c \leftarrow \{z \div \overline{c} \mid (s, \overline{c}, s') \in T_c \text{ and } (s', z) \in \mathcal{F}\}$
8: **if** $\exists (s, \overline{c}, s') \in T_c \cup T_u$ **then** $C \leftarrow C \cup \{(s, z) \mid z \in Z_u \sqcap Z_c\}$
9: **return** C

1: **function** $UZones(s, T_u, \mathcal{F})$
2: $zones \leftarrow \{(0^n; \infty^n)\}$
3: **for all** $(s, \overline{c}, s') \in T_u$ **do**
4: $zones \leftarrow zones \sqcap \{z \div \overline{c} \mid (s', z) \in \mathcal{F}\}$
5: **return** $zones$

Proof The set $Next(\mathcal{F})$ is either the intersection of Z_c and Z_u, the union, or the empty set. The set Z_c is finite, since $\mathcal{F}$ and T_c are finite sets and the size of Z_c is determined by the number zones in $\mathcal{F}$ and controllable transitions. The set Z_u is finite, since F and T_u are finite sets and the size of Z_u is determined by the number zones in $\mathcal{F}$ and uncontrollable transitions. Hence, since Z_c and Z_u are finite, so is $Next(\mathcal{F})$. □

Let $\mathcal{G} = (S, s_0, \mathcal{AP}, L, T_c, T_u)$ be an n-WGG then we have that $F \subseteq \mathcal{C}$ and we define the function $Add(F)$ s.t.

$$Add(F) = \left\{ (s, \overline{w}) \;\middle|\; \begin{array}{l} \text{whenever } s \overset{\overline{c}}{\dashrightarrow} s' \text{ then } (s', \overline{w} + \overline{c}) \in F \text{ and} \\ \text{if } s \rightarrow \text{ then } s \overset{\overline{c}}{\rightarrow} s' \text{ s.t. } (s', \overline{w} + \overline{c}) \in F \text{ and} \\ \text{there exists } (s, \overline{c}, s') \in T_c \cup T_u. \end{array} \right\}$$

We argue about the correctness of Algorithm 3 by showing that the semantics of $Next(\mathcal{F})$ corresponds to the set $Add([\![\mathcal{F}]\!])$.

Lemma 17
Let $\mathcal{G} = (S, s_0, \mathcal{AP}, L, T_c, T_u)$ be an n-WGG. Given $\mathcal{F} \subseteq \zeta$, Algorithm 3 constructs the set $Next(\mathcal{F})$ s.t.

$$[\![Next(\mathcal{F})]\!] = Add([\![\mathcal{F}]\!])$$

Proof Let $\mathcal{G} = (S, s_0, \mathcal{AP}, L, T_c, T_u)$ and $s \in S$. We prove that $(s, \overline{w}) \in [\![Next(\mathcal{F})]\!]$ iff. $(s, \overline{w}) \in Add([\![\mathcal{F}]\!])$.

$\Rightarrow$ Assume $(s, \overline{w}) \in [\![Next(\mathcal{F})]\!]$, then we want to show that $(s, \overline{w}) \in Add([\![\mathcal{F}]\!])$. Thus we need to prove that:

Whenever $s \overset{\overline{c}}{\dashrightarrow} s'$ then $(s', \overline{w} + \overline{c}) \in [\![\mathcal{F}]\!]$: There are two cases:

- $\nexists (s, \overline{c}, s') \in T_u$. Then by line 2 in $UZones$, $Z_u = \{(0^n; \infty^n)\}$ and as there are no uncontrollable transitions from s we simply return this set. This zone will not restrict the zones produced by controllable transitions in line 7 in $Next$.

– $\exists(s,\overline{c},s') \in T_u$. Thus if $\overline{w} \in [\![Z_u]\!]$ then for all $(s,\overline{c},s') \in T_u$ then $(s',\overline{w}+\overline{c}) \in [\![\mathcal{F}]\!]$. By line 2 in *UZones* we set $zones = \{(0^n;\infty^n)\}$. In line 3 and line 4 we go through every transition $(s,\overline{c},s') \in T_u$ and intersect *zones* with the zones $z' \div \overline{c}$ where $(s',z') \in \mathcal{F}$. Notice that if $(s',\overline{w}+\overline{c}) \in [\![\mathcal{F}]\!]$ then $\overline{w} \in [\![z' \div \overline{c}]\!]$. Also notice that if there is some uncontrollable transition $(s,\overline{c},s') \in T_u$ s.t. for all $(s',z') \in \mathcal{F}$ then $z' \div \overline{c} = \bot$ we have that $zones = \emptyset$. Lastly, if there are no $\overline{w} \in [\![zones]\!] \cap [\![z' \div \overline{c}]\!]$ we have that $zones = \bot$. Hence, if $\overline{w} \in [\![Z_u]\!]$ then for all $(s,\overline{c},s') \in T_u$ then $(s',\overline{w}+\overline{c}) \in [\![\mathcal{F}]\!]$.

If $s \rightarrow$ then $s \xrightarrow{\overline{c}} s'$ s.t. $(s',\overline{w}+\overline{c}) \in [\![\mathcal{F}]\!]$: There are two cases:

– $\nexists(s,\overline{c},s') \in T_c$. Then by line 5 of *Next*, $Z_c = \{(0^n;\infty^n)\}$. This zone will not restrict the zones produced by uncontrollable transitions in line 7 in *Next*.

– $\exists(s,\overline{c},s') \in T_c$. Thus if $\overline{w} \in [\![Z_c]\!]$ then there is some $(s,\overline{c},s') \in T_c$ s.t. $(s',\overline{w}+\overline{c}) \in [\![\mathcal{F}]\!]$. By line 6 in *Next* we go through all controllable transitions $(s,\overline{c},s') \in T_c$ and find the disjunction of all zones $z' \div \overline{c}$ where $(s',z') \in \mathcal{F}$. Again we notice that if $(s',\overline{w}+\overline{c}) \in [\![\mathcal{F}]\!]$ then $\overline{w} \in [\![z' \div \overline{c}]\!]$.

There exists $(s,\overline{c},s') \in T_c \cup T_u$: If $(s,\overline{w}) \in Next(\mathcal{F})$ then there exist some $(s,z) \in Next(\mathcal{F})$ where $\overline{w} \in [\![z]\!]$. Then by line 7 in $Next(\mathcal{F})$ there is some transition $(s,\overline{c},s') \in T_c \cup T_u$ and $z \in Z_u \sqcap Z_c$.

Thus whenever $(s,\overline{w}) \in [\![Next(\mathcal{F})]\!]$ then $(s,\overline{w}) \in Add([\![\mathcal{F}]\!])$.

$\Leftarrow$ Assume $(s,\overline{w}) \in Add([\![\mathcal{F}]\!])$, then we want to show that $(s,\overline{w}) \in [\![Next(\mathcal{F})]\!]$. By Definition of $Add([\![\mathcal{F}]\!])$ we have that the following holds:

Whenever $s \overset{\overline{c}}{\dashrightarrow} s'$ then $(s',\overline{w}+\overline{c}) \in [\![\mathcal{F}]\!]$: Whenever $s \overset{\overline{c}}{\dashrightarrow} s'$ then $(s',\overline{w}+\overline{c}) \in [\![\mathcal{F}]\!]$. Then we have that for all $(s,\overline{c},s') \in T_u$ there is some $(s',z') \in \mathcal{F}$ s.t. $\overline{w}+\overline{c} \in [\![z']\!]$. By line 4 in *UZones* we have that *zones* is the set remaining after intersecting all zones $z' \div \overline{c}$. In line 4 in *Next* we define Z_u as *zones*. Thus we can ensure that there is a zone $z'' \in Z_u$ s.t. $\overline{w} \in [\![z'']\!]$.

If $s \rightarrow$ then $s \xrightarrow{\overline{c}} s'$ s.t. $(s',\overline{w}+\overline{c}) \in [\![\mathcal{F}]\!]$: Then we have that if there is some transition $(s,\overline{c},s') \in T_c$ then we have that there exists $(s',z') \in \mathcal{F}$ s.t. $(s',\overline{w}+\overline{c}) \in [\![\mathcal{F}]\!]$. By line 6 in *Next* we have that Z_c is constructed by disjunction of all possible zones $z' \div \overline{c}$. Thus, there is some zone $z'' \in Z_c$ s.t. $\overline{w} \in [\![z'']\!]$

There exists $(s,\overline{c},s') \in T_c \cup T_u$: Then by line 7 in *Next* we add all symbolic configurations (s,z) where $z \in Z_c \sqcap Z_u$. As we know that there is a zone $z_u \in Z_u$ s.t. $\overline{w} \in [\![z_u]\!]$ and that there is a zone $z_c \in Z_c$ s.t. $\overline{w} \in [\![z_c]\!]$ then by Definition 5.10 then there is a zone $z \in Z_c \sqcap Z_u$ s.t. $\overline{w} \in [\![z]\!]$.

Hence there is some $(s,z) \in Next(\mathcal{F})$ s.t. $\overline{w} \in [\![z]\!]$. Thus, whenever $(s,\overline{w}) \in Add([\![\mathcal{F}]\!])$ then $(s,\overline{w}) \in [\![Next(\mathcal{F})]\!]$.

Hence, $(s,\overline{w}) \in [\![Next(\mathcal{F})]\!]$ iff. $(s,z) \in Next(\mathcal{F})$ s.t. $\overline{w} \in [\![z]\!]$. □

With the set of winning symbolic configurations reachable in a single step defined in Algorithm 3 we present Algorithm 4 which computes $\mathcal{F}_{final}$. The algorithm takes a ReachWCTL^u_l game as input and returns the set of winning symbolic configurations.

First Algorithm 4 calls Algorithm 2 to create $\mathcal{F}_0$. From there it repeatedly calls Algorithm 3 until the set of winning configurations stabilize.

We can now complete Example 5.3, with the finite zone representation.

Algorithm 4 Algorithm computing $\mathcal{F}_{final}$

Input: n-WGG $\mathcal{G} = (S, s_0, \mathcal{AP}, L, T_c, T_u)$ and ReachWCTL$_l^u$ formula $\varphi = AF\psi$
Output: A set of winning configurations C where $[\![C]\!] = \mathcal{F}_{final}$
1: **function** ReachAlgorithm($\mathcal{G}, \varphi$)
2: $i \leftarrow 0$
3: $\mathcal{F}_i \leftarrow \{(s, z) \mid z \in GetBasis(\psi, (0^n; \infty^n), s)$ and $s \in S\}$
4: **while** $\mathcal{F}_i \neq \mathcal{F}_{i-1}$ **do**
5: $i \leftarrow i + 1$
6: $\mathcal{F}_i \leftarrow \mathcal{F}_{i-1} \cup Next(\mathcal{F}_{i-1})$
7: **return** $\mathcal{F}_i$

Example 5.5 (Self Driving Car)
Consider our earlier example with the game $(\mathcal{G}, \varphi)$, where $\mathcal{G}$ is illustrated in Figure 10 and $\varphi = AF(\#1 \leq 3 \wedge \#2 \geq 5 \wedge destination)$, we now have a finite amount of configurations in $\mathcal{F}_0$, since GetBasis returns a finite set of zones for each state $s \in S$ thus,

$$\mathcal{F}_0 = \{(s_2, ((0,5);(3,\infty)))\}.$$

We compute each iteration using the function $Next$ and call the set of symbolic configurations $\mathcal{F}_i$, where i is the number of iterations.

$$\begin{aligned}
\mathcal{F}_1 &= \mathcal{F}_0 \cup \{(s_1, ((0,4);(2,\infty)))\} \cup \{(s_1, ((0,5);(1,\infty)))\} \\
\mathcal{F}_2 &= \mathcal{F}_1 \cup \{(s_1, ((0,1);(2,\infty)))\} \cup \{(s_1, ((0,2);(1,\infty)))\} \\
&\quad \cup \{(s_0, ((0,1);(0,\infty)))\} \\
\mathcal{F}_3 &= \mathcal{F}_2 \cup \{(s_1, ((0,0);(2,\infty)))\} \cup \{(s_1, ((0,0);(1,\infty)))\} \\
&\quad \cup \{(s_0, ((0,0);(0,\infty)))\} \\
\mathcal{F}_4 &= \mathcal{F}_3 \cup \emptyset
\end{aligned}$$

Lemma 18
Given a finite ReachWCTL$_l^u$ n-WG as input, Algorithm 4 terminates.

Proof We prove that the number of symbolic configurations, which can be created by repeated calls to $Next$ is finite. By Lemma 16 we have that $Next(\mathcal{F}_i)$ is a finite set. Now, observe that for any symbolic configuration $(s, (\bar{l}; \bar{u})) \in Next(\mathcal{F}_i)$ then we have that either $\bar{u}[i] \in \mathbb{N}_0$ or $\bar{u}[i] = \infty$. By definition of $\div$ we have that any component defined with ∞ will never change. As any symbolic configuration created by $Next$ will be the result of a subtraction and intersection, the set of symbolic configurations will eventually stabilize. Thus Algorithm 4 terminates. □

Theorem 19 (Decidability of Synthesis for ReachWCTL$_l^u$ Games)
The synthesis problem for ReachWCTL$_l^u$ games is decidable.

Proof We prove this by showing that Algorithm 4 returns the set representing all winning configurations and from that we can extract a winning strategy. By Lemma 14 we have that the winning symbolic configurations created in Algorithm 4 Line 3 represents all winning configurations which trivially satisfies the objective. By Lemma 17 we can use Lemma 6 and 7, and from those we get that Algorithm 4 returns a set of symbolic configurations representing the set of all winning configurations. By Lemma 18 we get that Algorithm 4 terminates. Extracting the strategy is simply a manner of slightly modi-

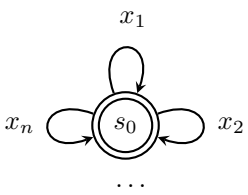
fying the strategy extraction method used in Section 5.1. Thus we have that the synthesis problem for ReachWCTL^u_l games is decidable. □

Lemma 20
The synthesis problem for ReachWCTL^u_l games is NP-hard.

Proof We reduce the NP-complete SUBSET-SUM problem [17] into the reachability synthesis problem. The SUBSET-SUM problem states that given a set of integers $X = \{x_1, \ldots, x_n\} \subseteq \mathbb{N}$ and an integer $t \in \mathbb{N}$ is there a subset (or multiset) $I = \{i_1, \ldots, i_n\} \subseteq X$ s.t.

$$\sum_{j=1}^{n} i_j = t \ ?$$

Given an instance of the SUBSET-SUM problem, we construct a reachability game $(\mathcal{G}, \varphi)$ where $\mathcal{G} = (S, s_0, \mathcal{AP}, L, T_c, T_u)$ is the 1-WGG illustrated in Figure 13a s.t. $T_u = \emptyset$ and for every $x \in X$ there is a edge $s \xrightarrow{x} s \in T_c$. The formula φ is shown in Figure 13b.



(a) n-WGG $\mathcal{G}$

$$\varphi = AF(\#1 = t)$$

(b) ReachWCTL^u_l formula φ

Figure 13: The game $(\mathcal{G}, \varphi)$ where $\mathcal{G}$ is an n-WGG and φ is an ReachWCTL^u_l formula.

Now, consider the reachability synthesis problem for the game $(\mathcal{G}, \varphi)$ of whether there is a winning SSC strategy σ s.t. $\mathcal{G}\restriction\sigma, s_0 \vDash_{0^n} \varphi$. It is clear that there is solution to the reachability synthesis problem if and only if there is a solution to the SUBSET-SUM problem.□

Theorem 21
The synthesis problem for ReachWCTL^u_l games belongs to EXPTIME.

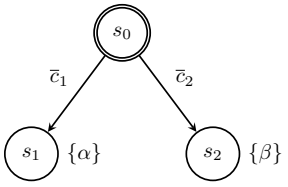
Proof By Theorem 19 we have that Algorithm 4 solves the synthesis problem for ReachWCTL^u_l games. We now prove that Algorithm 4 runs in exponential time. Obviously, GetBasis has polynomial time complexity in the size of the input.

Let $(\mathcal{G}, \varphi)$ be a ReachWCTL^u_l game where $\mathcal{G} = (S, s_0, \mathcal{AP}, L, T_c, T_u)$ is an n-WGG, $T = T_c \cup T_u$ and $\varphi = AF\psi$. Let $k = max(\{c \mid c \bowtie \#i \in Sub(\varphi)\})$ be the largest constant compared against any component in the formula φ. Notice that $|GetBases(\psi, (0^n; \infty^n), s_0)| \leq |\psi|$, and therefore $|\mathcal{F}_0| \leq |S| \cdot |\psi|$, hence computing $\mathcal{F}_0$ can be done in polynomial time. Now, computing $Next(\mathcal{F}_i)$, for any $i \in \mathbb{N}_0$, takes $O(|S| \times |T| \times |\mathcal{F}_i|)$ time and $|Next(\mathcal{F}_i)| \leq |S| \times |T| \times |\mathcal{F}_i|$. The size of $\mathcal{F}_i$ is determined by $|S|$ and the number of vectors smaller than k^n, thus we have that $|\mathcal{F}_i| \leq |T| \cdot |S| \cdot k^{2n}$. This also gives us a maximum number of calls to $Next$. We can now conclude that Algorithm 4 runs in $O(|T|^2 \cdot |S|^2 \cdot k^{4n})$ which is exponential in the number of dimension in the game graph. □

6 Synthesis for Constant Bound WCTL Games

In this section we discuss the synthesis problem for cb-WCTL games and provide an argument for why a simple extension of the method presented in Section 5 is not possible.

Given Algorithm 3 it is trivial to use Algorithm 4 to solve the synthesis problem for a game defined with non-nested cb-WCTL formula. However, when we allow both conjunction and nesting we can get the game shown in Figure 14.



(a) $\mathcal{G} = (S, s_0, \mathcal{AP}, L, T_c, T_u)$

$$\varphi = AF(\alpha) \wedge AF(\beta)$$

(b) cb-WCTL formula φ

Figure 14: cb-WCTL game where the synthesis method from Section 5.2 produces an invalid strategy.

Intuitively, a solution would simply be to find all winning symbolic configurations for the ReachWCTL^u_l games $(\mathcal{G}, AF(\alpha))$ and $(\mathcal{G}, AF(\beta))$ and check if the intersection of these sets yielded any symbolic configurations of the form $(s_0, (0^n; \overline{u}))$. But on closer inspection of the example, we see that the output of Algorithm 4 would be the following for each game:

$$\text{REACHALGORITHM}((\mathcal{G}, AF(\alpha))) = \{(s_0, (0^n; \infty^n)), (s_1, (0^n; \infty^n))\}.$$
$$\text{REACHALGORITHM}((\mathcal{G}, AF(\beta))) = \{(s_0, (0^n; \infty^n)), (s_2, (0^n; \infty^n))\}.$$

Thus given a simple intersection of the sets we would get the following:

$$\{(s_0, (0^n; \infty^n)), (s_1, (0^n; \infty^n))\} \sqcap \{(s_0, (0^n; \infty^n)), (s_2, (0^n; \infty^n))\} = \{(s_0, (0^n; \infty^n))\}.$$

This would suggest that there exist a winning strategy for the game. But obviously there cannot be a strategy σ s.t. $\mathcal{G}\restriction\sigma, s_0 \vDash_{0^n} \varphi$ as one of the controllable transitions would not be present in $\mathcal{G}\restriction\sigma$, and both are required to satisfy $AF(\alpha) \wedge AF(\beta)$. Based on this we conclude that another approach is needed to solve the synthesis problem for cb-WCTL games.

7 Conclusions

We considered the formalism n-WKS, which is a Kripke structure extended with multiple non-negative weights. We proved that the model checking problem of WCTL on a finite 3-WKS is undecidable. However, we found that removing the possibility of subtracting two dimensions from each other and removing boolean comparison between two dimensions was sufficient to make a decidable sub-logic. We called this sub-logic cb-WCTL and proved that the model checking problem of cb-WCTL on a finite n-WKS is decidable. Adding negative weights in the game formalism would be an interesting subject for future work. Because of the increased expressiveness of the model, a new, more restrictive, sub-logic is needed, as the n-WKS can now simulate the counters, and cb-WCTL can simulate the zero check, thus cb-WCTL on an n-WKS with negative weights is undecidable.

We presented a framework for multi weighted two player games based on the n-WKS formalism and the cb-WCTL. This framework was developed during our pre-specialization project, where we investigated multiple types of strategies. In this thesis we investigated synthesis of a sub-logic of cb-WCTL for specifying weighted reachability objectives, called

ReachWCTL$_l^u$. ReachWCTL$_l^u$ expresses reachability objectives with the possibility of both upper- and lower-bounds on all components.

The synthesis method is first presented for ReachWCTL^u games which are solvable in pseudo-polynomial time. Additionally we show that if there is a winning strategy there is a winning finite memory strategy (SSC strategy); which only remembers the current state and the cost of a run. For ReachWCTL$_l^u$ games we utilizes a symbolic representation of the vector state space, in order to represent infinite sets of vectors. As with ReachWCTL^u games we have that if there is a winning strategy for a ReachWCTL$_l^u$ game, then there is winning SSC strategy. Furthermore, we provide a lower bound on the complexity of synthesis of ReachWCTL$_l^u$ games which is NP-Hard and also provide an upper-bound in the form of an algorithm which runs in exponential time. Lastly, we have also shown that synthesis of cb-WCTL games cannot be achieved by using the method presented in this thesis in a component wise manner based on the nested construction of the game objective. In particular, the combination of nesting temporal operators and conjunctions requires a different approach.

There are several areas worth considering as future work, both practical and theoretical. In relation to possible application of these synthesis algorithms, it still remains to develop a prototype and provide a translation from a winning strategy to an actual implementation of a controller. Furthermore, on-the-fly algorithms have shown, in the context of model checking, to significantly improve performance although the theoretical complexity is not improved. We believe that this could be achieved using dependency graphs, where there has been recent advances with on-the-fly fixed-point algorithms [7].

The subject of probability is also relevant for two reasons. First, it allows the modelling of non-cooperative players in the environment, in contrast to the antagonistic opponent. Second, there might be application areas where no winning strategy exist, but there is a high probability of success. E.g. Recall the self driving car, it may not be possible to have a winning strategy due to the immense number of cars on the roads, but based on road data one might be able to predict with high certainty that the strategy succeed. With probabilistic models we could concoct strategies in an environment, where victory cannot be ensured, but is probable, to some degree.

8 Bibliographical Remarks

This thesis is partly based on our pre-specialization project from fall 2016 [11]. Section 2 is a concise version of Chapter 2 and Section 3.1 from [11] with minor corrections. The undecidability result presented in Theorem 1 in Section 3 is inspired by the undecidability proof from Theorem 1 in Chapter 3 of [11]. However, it proves a stronger result as it shows undecidability on a 3-WKS instead of a 4-WKS. Section 4 is a concise version of Sections 3.2 and 3.3 from [11] with minor corrections, with the exception of Proposition 3 which is a new result.

References

[1] Patricia Bouyer, Kim Guldstrand Larsen, and Nicolas Markey. “Model Checking One-Clock Priced Timed Automata.” In: *Logical Methods in Computer Science* 4.2 (July 28, 2009).

[2] J. Richard Büchi and Lawrence H. Landweber. “Definability in the Monadic Second-Order Theory of Successor”. In: vol. 34. 2. Association for Symbolic Logic, 1969, pp. 166–170.

[3] Franck Cassez, Alexandre David, Emmanuel Fleury, Kim G Larsen, and Didier Lime. "Efficient on-the-fly algorithms for the analysis of timed games". In: *International Conference on Concurrency Theory*. Springer. 2005, pp. 66–80.

[4] Krishnendu Chatterjee, Laurent Doyen, Thomas A Henzinger, and Jean-François Raskin. "Generalized mean-payoff and energy games". In: *arXiv preprint arXiv:1007.1669* (2010).

[5] Alonzo Church. "Logic, arithmetic and automata". In: *Proceedings of the international congress of mathematicians*. 1962, pp. 23–35.

[6] Edmund M. Clarke and E. Allen Emerson. "Design and synthesis of synchronization skeletons using branching time temporal logic". In: *Logics of Programs: Workshop, Yorktown Heights, New York, May 1981*. Springer Berlin Heidelberg, 1982, pp. 52–71.

[7] A.E. Dalsgaard, S. Enevoldsen, P. Fogh, L.S. Jensen, T.S. Jepsen, I. Kaufmann, K.G. Larsen, S.M. Nielsen, M.Chr. Olesen, S. Pastva, and J. Srba. "Extended Dependency Graphs and Efficient Distributed Fixed-Point Computation". In: *Proceedings of the 38th International Conference on Application and Theory of Petri Nets and Concurrency (Petri Nets'17)*. LNCS. To appear. Springer-Verlag, 2017, pp. 1–20.

[8] Jonas Finnemann Jensen, Kim Guldstrand Larsen, Jiří Srba, and Lars Kaerlund Oestergaard. "Efficient model-checking of weighted CTL with upper-bound constraints". In: *International Journal on Software Tools for Technology Transfer* (2014), pp. 1–18.

[9] Barbara Jobstmann and Roderick Bloem. "Optimizations for LTL synthesis". In: *2006 Formal Methods in Computer Aided Design*. IEEE. 2006, pp. 117–124.

[10] Marcin Jurdziński, Ranko Lazić, and Sylvain Schmitz. "Fixed-Dimensional Energy Games are in Pseudo-Polynomial Time". In: *Automata, Languages, and Programming: 42nd International Colloquium, ICALP 2015, Kyoto, Japan, July 6-10, 2015, Proceedings, Part II*. Ed. by Magnús M. Halldórsson, Kazuo Iwama, Naoki Kobayashi, and Bettina Speckmann. Berlin, Heidelberg: Springer Berlin Heidelberg, 2015, pp. 260–272.

[11] Isabella Kaufmann, Lasse S. Jensen, and Søren M. Nielsen. "Controller Synthesis By Solving Multi Weighted Games". In: *9. Smester*. Aalborg University. 2016.

[12] Saul A. Kripke. "Semantical Considerations on Modal Logic". In: *Acta Philosophica Fennica* 16.1963 (1963), pp. 83–94.

[13] Orna Kupferman and Moshe Y. Vardi. "Safraless Decision Procedures". In: *Proceedings of the 46th Annual IEEE Symposium on Foundations of Computer Science*. FOCS '05. Washington, DC, USA: IEEE Computer Society, 2005, pp. 531–542. ISBN: 0-7695-2468-0.

[14] Orna Kupferman and Moshe Y Vardi. "Synthesis with incomplete information". In: *Advances in Temporal Logic*. Vol. 16. 2000, pp. 109–127.

[15] Xinxin Liu and Scott A Smolka. "Simple linear-time algorithms for minimal fixed points". In: *International Colloquium on Automata, Languages, and Programming*. Springer. 1998, pp. 53–66.

[16] Marvin L. Minsky. *Computation: Finite and Infinite Machines*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1967.

[17] Robert A Proctor. "Solution of two difficult combinatorial problems with linear algebra". In: vol. 89. 10. JSTOR, 1982, pp. 721–734.

[18] Wolfgang Thomas. "On the Synthesis of Strategies in Infinite Games". In: *STACS 95*. Springer. 1995, pp. 1–13.