# Controller Synthesis

- By Solving Multi Weighted Games -

Project Report

des902e16

Aalborg University
Department of Computer Science

AALBORG UNIVERSITY
STUDENT REPORT

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

**Title:**
Controller Synthesis By Solving Multi Weighted Games

**Theme:**
Synthesis

**Project Period:**
Fall Semester 2016

**Project Group:**
des902e16

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

**Supervisor(s):**
Kim G. Larsen
Jirí Srba

**Copies:** 0

**Page Numbers:** 54

**Date of Completion:**
December 20, 2016

**Abstract:**

Controller synthesis is investigated through a game theoretic view. In the context of games, synthesis is the extraction of a winning strategy. We investigate multi-weighted games, and various strategy types. We hierarchically order these strategy types by expressiveness. We also present a weighted computation tree logic, for which we provide an undecidability result, and a decidable sublogic, limited to reachability with upper bounds. Further more we provide complexity results for the synthesis problem with the reachability sublogic. We present two methods for synthesis of strategies in 1-weighted games: We extend the attractor set method with weights giving a global algorithm with polynomial time complexity. And we also show how a strategy can be extracted using prefixed-point assignment of a symbolic dependency graph.

# Contents

# Preface

This project is the first part of our Master's Thesis from the Department of Computer Science at Aalborg University. It is written during the fall of 2016 and our future works describe topics we plan to cover in the second part of our Master's Thesis in the spring of 2017. We would like to thank our supervisors Jiří Srba and Kim Guldstrand Larsen for the feedback they have given throughout the project.

Aalborg University, December 20, 2016

Isabella Kaufmann
<ikaufm12@student.aau.dk>

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

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

# Chapter 1

# Introduction

Synthesis is the construction of a correct implementation based on logical specifications. The synthesis problem was first formulated by Church in 1962 [5], focusing on whether the solution of a specification, formulated in monadic second-order logic, can be algorithmically generated. This problem was proven decidable by Büchi and Landweber in 1969 [3]. Following this result the problem has been continually expanded.

Controller synthesis has for several years been studied under a game theoretic formulation [12]. The construction of a controller acting in an environment can also be viewed as finding and extracting a strategy in a game. In 1998 Pnueli et al. provided a synthesis algorithm for safety games and timed safety game [12]. Other examples include synthesis for LTL by Pnueli and Rosner [13] and synthesis of synchronisation skeletons for CTL by Emerson and Clarke [6]. Recently the focus in this area has been on expanding the synthesis problem towards more complex real life applications, as well as providing algorithms efficient enough to have some practical value. There has been a great interest in verification of models with continuous time such as the timed automata, however the free flow of time has proven difficult to solve. In 2009 Bouyer, Larsen, and Markey showed that properties expressed in CTL and Weighted Computation Tree Logic (WCTL) is PSPACE-complete, with respect to a one-clock timed automata, and they become undecidable with three or more clocks [1].

In this paper we focus on synthesis of specifications with bounds on resource consumption. Allowing only discrete representation of time and other resources, in the hopes of achieving results with practical application. We introduce a game formalism called an $n$-Weighted Game ($n$-WG). An $n$-WG is a two player game where a player's moves (choice of transition in the underlying model) cause a change in resource levels. The underlying model of the $n$-WG is an $n$-Weighted Kripke Structure ($n$-WKS); an extension of the Kripke structure, with arbitrary many weights on transitions. With this type of game, we can simulate the behavior of two player's

(The controller and its environment) acting and influencing each other simultaneously. This will allow us to model real life problems and synthesize strategies to efficient solutions.

To clarify which type of problems this kind of game can solve we give a simple example. Given a game graph which models a (knowledge) hungry student, we can illustrate choices made in accordance to the current situation as environmental, and choices free of circumstance as controllable. Consider the game graph illustrated in Figure 1.1.

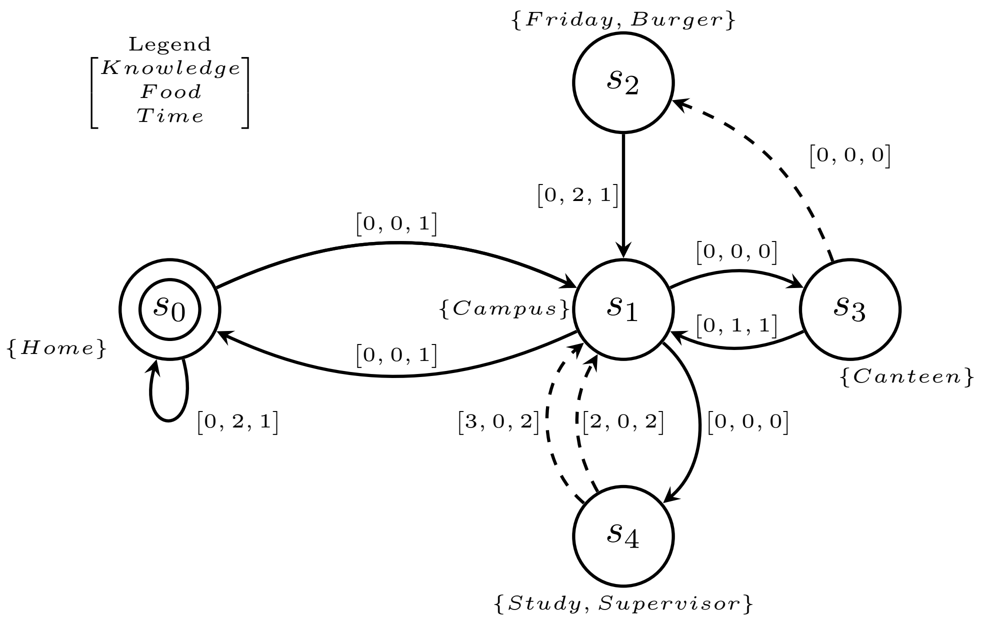


**Figure 1.1:** 3-weighted game graph G where the first vector component models the knowledge obtained by a student, the second her food intake, and the third component represents time elapsed.

In this game graph the choices are modeled as transitions which have a number of weights to illustrate resource consumption and accumulation. The controllers actions are depicted by the whole lines and the environments by the dashed lines.

Say a student needs at least 4 food units to get through the day and she can spend time gaining knowledge, either by meeting with her supervisor or studying. As a day only has so many hours, we can now specify the problem as *"Can a student obtain 8 knowledge units within 16 time units, if she has to eat at least 4 food units within the same time frame?"*. By synthesizing a winning strategy in this game, we can quickly determine how to solve this problem if there exist a solution. In the context of Figure 1.1, the controller is the decision maker, when it comes to the students actions, and the environments actions are phenomenons which the student has no control over. E.g. If a supervisor is available or if it is Friday. In Figure 1.1 there is a strategy for achieving 8 units of knowledge and 4 units of food within 16 time units. Consider this procedure of choices the student might make:

```
1: if $Home \wedge (Food = 0)$ then goto $s_0$
2: else if $Home$ then goto $s_1$
3: else if $Campus \wedge (Food < 4)$ then goto $s_3$
4: else if $Friday \wedge Burger$ then goto $s_1$
5: else if $Canteen$ then goto $s_1$
6: else goto $s_4$
```

The procedure is a strategy for the student to follow and if the strategy leads to achieving the goal, we call it a winning strategy; the above is such a strategy. Consider the student using the strategy. She starts at home on a Thursday with all weights set to 0.

1. $(0,0,0)$. First she eats
2. $(0,2,1)$. She leaves for campus.
3. $(0,2,2)$. She goes to the canteen, but it is a Thursday, so she eats a regular meal.
4. $(0,3,3)$. She goes to the canteen again as she is still hungry.
5. $(0,4,4)$. She can now start studying, and her supervisor is available.
6. $(3,4,6)$. Her supervisor is now at another meeting, but she still studies.
7. $(5,4,8)$. The supervisor can now give feedback to her work.
8. $(8,4,10)$. The student has now obtained 8 units of knowledge and eaten 4 units of food, within 16 time units.

Notice, that the environment might choose to act differently, however this will only help the student achieve her goal faster. For example, on Fridays the food is burgers, giving double the food in the same time frame. The choice to get a burger the dashed transition from $s_3$ to $s_2$. This means that the environment decides if the student can have a burger. Despite this the student can get enough food in time, having the regular canteen meals. It is the synthesis of such strategies, that satisfies the properties in either case we investigate in this paper.

## 1.1 Related Work

Synthesis has been researched in different contexts. In 2009 Thomas [14] presented the *automaton theoretic framework*, defining a generalised two player turn-based game comprised of: a game graph, a winning condition and a strategy. We extend this framework to two player game and weights on player actions (transitions in the game).

In 2008 Bouyer et al. [2] studied games both in a timed and un-timed setting. Additionally they look at both negative and positive weights in the pursuit of finding infinite strategies that keep the accumulated weight of a resource in some interval. In 2011 Fahrenberg et al. [7] explored the same domain, with multiple weights in the formalism. In contrast we do not consider timed games and only look at nonnegative weights. However, we consider the possibility of modeling multiple resources, and specify a more expressive logic to express the desired properties.

In 2006 Jobstmann and Bloem [10] presented a polynomial time method for synthesising strategies in the context of co-Büche tree automatons and the full linear temporal logic. In contrast we specify winning conditions in CTL variants with multiple weights. Bouyer, Larsen, and Markey [1] showed that model checking of CTL, with respect to a Weighted Timed Automata (WTA), to be decidable with 1 clock and undecidable with 3 clocks (2 clocks remain unknown). In 2005 Cassez et al. [4], investigated an on-the-fly approach for synthesis of reachability and safety properties for Timed Automata (TA) based games. In this article, we use a game model without continuous time, and instead extend with multiple weights, allowing discretion of time and other resources.

Cassez et al. [4] used the Dependency Graph (DG) framework first presented by Liu and Smolka [11] in 1998. They developed the DG framework with the intention of solving the HORNSAT problem. The DG framework has been extended since then, and used to solve other problems than HORNSAT. In 2014 Jensen et al. [9] presented the Symbolic Dependency Graph (SDG) framework, which were used to model check WCTL[1] properties of Weighted Kripke Structure (WKS) with 1 weight. In this paper we show how the SDG framework can be used to synthesize strategies for a subset of WCTL.

## 1.2 Outline

In this paper we define the notion of games, and how computing strategies for games relates to synthesis of controllers. In Chapter 2 we present the preliminary formalities used throughout the paper; including $n$-Weighted Kripke Structure ($n$-WKS), WCTL and $n$-Weighted Games. We also define four types of strategies with different memory usage and define their relation to the synthesis problem. In Chapter 4 we present two methods for synthesizing strategies for 1-weighted reachability games, the attractor method and SDG framework. Lastly we define the encoding af a 1-weighted reachability game as a symbolic dependency graph

[1]The WCTL is a subset of the WCTL presented in this article.

and show a synthesis algorithm using this construction. Chapter 5 concludes the paper and gives an overview of topics for future work.

# Chapter 2

# Preliminaries

In this chapter we present the basic formalism and notation used throughout the paper; starting with the $n$-WKS a Kripke structure with arbitrary many weights on transitions. Next we define WCTL over $n$-WKS and show that the full WCTL is undecidable, based on this we then define a decidable subset of the logic, namely Reachability CTL with Upper-bounds (ReachWCTL$^u$). Following this the $n$-WG is presented and with it the notion of a strategy. We define four types of strategies and show how they relate to different sub-logics of WCTL. Lastly we introduce the synthesis problem and its relation to model checking when extracting and verifying a winning strategy.

## 2.1 $n$-Weighted Kripke Structure

In this section we present the $n$-Weighted Kripke Structure ($n$-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 Structure)**
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,
- $T \subseteq S \times \mathbb{N}_0^n \times S$ is a transition relation, with a weight vector of length $n$.

When $(s, \overline{w}, s') \in T$, where $s, s' \in S$ and $\overline{w} \in \mathbb{N}_0^n$ is a vector, then we write $s \xrightarrow{\overline{w}} s'$. When $s'$ is reachable from $s$, by any number of transitions, we write $s \to^* s'$ 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. An example of a finite 2-WKS is illustrated in Figure 2.1.

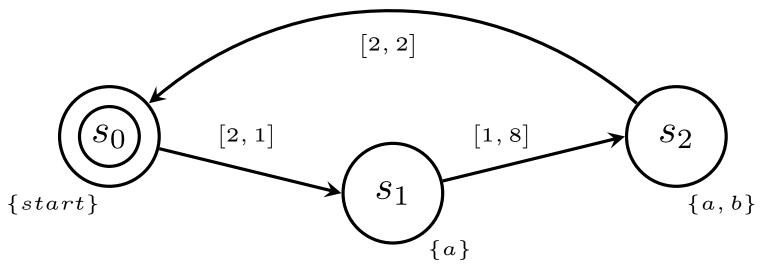


**Figure 2.1:** Example of an $n$-WKS. The initial state $s_0$ is marked with a double circle and $a, b, start \in \mathcal{AP}$

Let $\overline{w} \in \mathbb{N}_0^n$ be a vector of length $n$. We denote the $i$-th component of $\overline{w}$ by $\overline{w}[i]$, where $1 \leq i \leq n$. To set the $i$-th component of $\overline{w}$ to a specific value $k \in \mathbb{N}_0$ we write $\overline{w}[i \rightarrow k]$.

**Definition 2.2 (Ordering on Vectors)**
Let $\overline{w} = (\overline{w}[1], \ldots, \overline{w}[n])$ and $\overline{w}' = (\overline{w}'[1], \ldots, \overline{w}'[n])$ be vectors of length $n$, we write $\overline{w} \leq \overline{w}'$ iff $\overline{w}[i] \leq \overline{w}'[i]$ for all $i$ where $1 \leq i \leq n$.

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

$$\rho = s_0 \xrightarrow{\overline{w}_0} s_1 \xrightarrow{\overline{w}_1} s_2 \xrightarrow{\overline{w}_2} \ldots$$

where $s_i \xrightarrow{\overline{w}_i} s_{i+1}$ for all $i \geq 0$. Given a position $i \in \mathbb{N}_0$ along $\rho$, let $\rho(i) = s_i$, and $\textsc{Last}(\rho)$ be the last position along $\rho$, if $\rho$ is finite. We also define the concatenation operator $\circ$, s.t. if $\rho = s_0 \rightarrow^* s_n$ then $\rho \circ (s_n \xrightarrow{\overline{w}_n} s_{n+1}) = (s_0 \rightarrow^* s_n \xrightarrow{\overline{w}_n} s_{n+1})$.

We write the set of all runs $\rho$ in the $n$-WKS $K$ of the form $(\rho = s_0 \xrightarrow{\overline{w}_0} s_1 \xrightarrow{\overline{w}_1} \ldots)$ as $\Pi_K$. Furthermore we write the set of all finite runs $\rho$ in the $n$-WKS $K$ of the form $(\rho = s_0 \xrightarrow{\overline{w}_0} \ldots \xrightarrow{\overline{w}_{n-1}} s_n)$ as $\Pi_K^{fin}$. Lastly, we define $\Pi_K^{Max}$ as the set of all runs $\rho$ s.t. $\rho$ is infinite or $\textsc{Last}(\rho)$ is in a deadlock s.t. $\textsc{Last}(\rho) \not\rightarrow$.

**Remark 1**
In this paper we utilize standard vector operations when adding and subtracting the cost of a run.

**Definition 2.3 (Cost)**
Let $K = (S, s_0, \mathcal{AP}, L, T)$ be an $n$-WKS and $(\rho = s_0 \xrightarrow{\overline{w}_0} s_1 \xrightarrow{\overline{w}_1} \ldots)$ be a run in $K$. The cost of $\rho$, at position $i \in \mathbb{N}_0$, is then defined as:

$$\textsc{Cost}_\rho(i) = \begin{cases} 0^n & \text{if } i = 0 \\ \sum\limits_{i=0}^{i-1} \overline{w}_i & \text{otherwise,} \end{cases}$$

if $\rho$ is finite, we denote $\textsc{Cost}_\rho(\textsc{Last}(\rho))$ as $\textsc{Cost}(\rho)$.

# Chapter 3

# *n*-Weighted Games and Strategies

We begin this chapter with the definition of Weighted Computation Tree Logic (WCTL) defined in relation to *n*-WKSs. We prove that model checking of this logic is undecidable and define a decidable subset Reachability CTL with Upper-bounds (ReachWCTL$^u$). We then extend the automaton theoretic framework, presented by Thomas [14], to *n*-Weighted Games (*n*-WGs) and define four types of strategies for this type of game. The expressiveness of each type is then hierarchically determined and we prove that for a specific type of game, the reachability game, that there is a winning Single-State Cost Strategy (SSC strategy) if there exists a winning strategy. Lastly we give a short example of the correlation between *n*-WGs and strategies.

## 3.1 Weighted Computation Tree Logic

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

**Syntax**

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

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

**Semantics**

We define the semantics for a minimal set of operators as:

| | |
|---|---|
| $K, s \models_{\overline{w}} \textsc{true}$ | |
| $K, s \models_{\overline{w}} a$ | if $a \in L(s)$ |
| $K, s \models_{\overline{w}} \neg\varphi$ | if $s \not\models_{\overline{w}} \varphi$ |
| $K, s \models_{\overline{w}} \varphi_1 \vee \varphi_2$ | if $s \models_{\overline{w}} \varphi_1$ or $s \models_{\overline{w}} \varphi_2$ |
| $K, s \models_{\overline{w}} E\varphi_1 U\varphi_2$ | if there exists $(\rho = s \xrightarrow{\overline{w}_0} s_1 \xrightarrow{\overline{w}_1} s_2 \dots) \in \Pi_K^{Max}$ and a position $i \geq 0$ such that $K, \rho(i) \models_{\overline{w}+cost_\rho(i)} \varphi_2$ and $K, \rho(j) \models_{\overline{w}+cost_\rho(j)} \varphi_1$ for all $j < i$ |
| $K, s \models_{\overline{w}} A\varphi_1 U\varphi_2$ | if for all $(\rho = s \xrightarrow{\overline{w}_0} s_1 \xrightarrow{\overline{w}_1} s_2 \dots) \in \Pi_K^{Max}$, there is a position $i \geq 0$ such that $K, \rho(i) \models_{\overline{w}+cost_\rho(i)} \varphi_2$ and $K, \rho(j) \models_{\overline{w}+cost_\rho(j)} \varphi_1$ for all $j < i$ |
| $K, s \models_{\overline{w}} EX\ \varphi$ | if there is a state $s'$ such that $s \xrightarrow{\overline{w}'} s'$, and $K, s' \models_{\overline{w}+\overline{w}'} \varphi$ |
| $K, s \models_{\overline{w}} reset\ \#i$ in $\varphi$ | if $K, s \models_{\overline{w}[i \to 0]} \varphi$ |
| $K, s \models_{\overline{w}} \psi_1 \bowtie \psi_2$ | if $eval_{\overline{w}}(\psi_1) \bowtie eval_{\overline{w}}(\psi_2)$ |

The evaluation of $\psi$ is:

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

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 $\varphi$ the model checking problem is the question of whether $K, s \models_{\overline{w}} \varphi$ where $s \in S$ and $\overline{w}$ is vector of length $n$. When the $n$-WKS $K$ is obvious from context and the vector $\overline{w}$ is $0^n$ we simply write $s \models \varphi$.

**Theorem 1 (Undecidability of WCTL)**
The model checking problem for WCTL is undecidable on a finite $n$-Weighted Kripke Structure (finite $n$-WKS).

**Proof.** We use reduction from the halting problem for 2-counter machines. 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 4-WKS where whenever $K$ is in state $s_i$ then $M$ is in $Ins_i$. We use the vector of length four to increase and decrease the value of the counters, by the cost of a run $\rho \in \Pi_K^{fin}$ s.t.

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

$$C_2 = \textsc{Cost}_\rho(i)[2] - \textsc{Cost}_\rho(i)[4],$$

where $0 \leq i$ is the position of $s_i$ in the run $\rho$. We now construct $K$ and the formula $\varphi$ s.t. $M$ will halt for the empty input $(C_1 = 0, C_2 = 0)$ iff $s_0 \models \varphi$. In the simulation, the instructions are translated as seen in Figure 3.1 s.t. halt is simulated in Figure 3.1a, Increment in Figure 3.1b and 3.1c and Decrement in Figure 3.1d and Figure 3.1e.

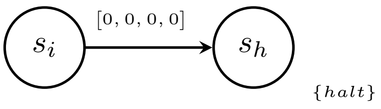
We then construct the formula,

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

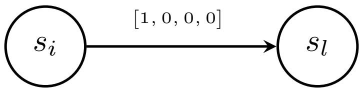
where

$$\begin{aligned} A &:= \{not\_zero\_1\} \Rightarrow \#1 > \#3 \wedge \{not\_zero\_2\} \Rightarrow \#2 > \#4 \\ B &:= \{is\_zero\_1\} \Rightarrow \#1 = \#3 \wedge \{is\_zero\_2\} \Rightarrow \#2 = \#4 \end{aligned}$$

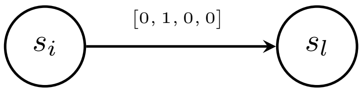
The formula $\varphi$ ensures that $M$ is simulated faithfully, as the counters are calculated in $\varphi$. When encountering the decrement rules, $\varphi$ enforces that the correct choice is taken, as the next state of the path will never satisfy both pre-conditions $A$ and $B$ if it is not allowed in $M$. To simulate the empty input, the weight vector $\overline{w} = 0^n$. As $M$ is faithfully simulated, we have that if $K, s_0 \models_{0^n} \varphi$ then $M$ will halt. We also 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 $\rho$ s.t. $\textsc{Last}(\rho) = s_h$ and for all $0 \leq i$ s.t. $\rho(i) \neq \textsc{Last}(\rho)$ it holds that $K, \rho(i) \models A \wedge B$, hence if $M$ halts, then $K, s_0 \models_{0^n} \varphi$. $\square$



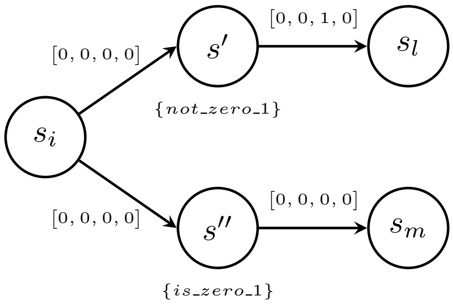
**(a)** Halt rule.



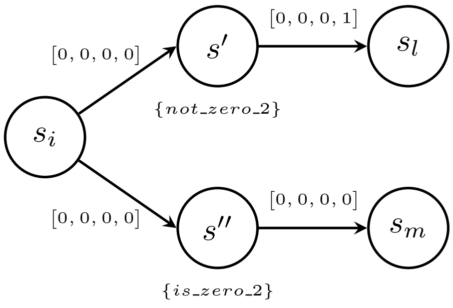
**(b)** Increment rule for $C_1$.



**(c)** Increment rule for $C_2$.



**(d)** Decrement rule for $C_1$.



**(e)** Decrement rule for $C_2$.

**Figure 3.1:** *n*-WKS simulation of a 2CM

### 3.1.1 Reachability CTL with Upper-bounds

In this section we present ReachWCTL$^u$, a sub-logic of the WCTL from Section 3.1.

#### Syntax

Let $K = (S, s_0, \mathcal{AP}, L, T)$ be an $n$-WKS. We then define the ReachWCTL$^u$ syntax, over $K$, as follows.

$$\begin{aligned}\varphi &:= EF_{\leq \bar{c}}\psi \mid AF_{\leq \bar{c}}\psi \\ \psi &:= a \mid \psi_1 \vee \psi_2 \mid \psi_1 \wedge \psi_2\end{aligned}$$

where $a \in \mathcal{AP}$ and $\bar{c} \in \mathbb{N}^n_\infty$.

#### Semantics

We define the semantics in relation to the logic presented in Section 3.1. The left-hand side is a formula in ReachWCTL$^u$ while the right-hand is a formula in WCTL. Following is the semantics for ReachWCTL$^u$:

$$\begin{aligned}AF_{\leq \bar{c}}\psi &\equiv A(\textsc{true})U((\#1 \leq \bar{c}[1] \wedge \cdots \wedge \#n \leq \bar{c}[n]) \wedge \psi) \\ EF_{\leq \bar{c}}\psi &\equiv E(\textsc{true})U((\#1 \leq \bar{c}[1] \wedge \cdots \wedge \#n \leq \bar{c}[n]) \wedge \psi)\end{aligned}$$

**Theorem 2 (Decidability of ReachWCTL$^u$)**
The model checking problem for ReachWCTL$^u$ is decidable on a finite $n$-WKS.

**Proof.** Let $K = (S, s_0, \mathcal{AP}, L, T)$ be an $n$-WKS and $\varphi$ be a ReachWCTL$^u$ formula and $s_0 \vDash \varphi$. Then there are two cases:

$\varphi = EF_{\leq \bar{c}}\psi$ Assume that given a run $\rho \in \Pi_K$ s.t. $\rho(i) \vDash \psi$ and $\textsc{Cost}_\rho(i) \leq \bar{c}$, then there is an acyclic finite run $\rho'$ where $\textsc{Last}(\rho') \vDash \psi$, $\textsc{Cost}(\rho') \leq \bar{c}$. As $S$ is finite there are finitely many acyclic runs, hence $\varphi$ is decidable by brute force enumeration of all finite acyclic runs.

$\varphi = AF_{\leq \bar{c}}\psi$ Assume that there exists a run $\rho \in \Pi_K^{Max}$ s.t. for all $0 \leq i$ we have that $\rho(i) \nvDash \psi$. Then $\rho$ is either acyclic and therefore finite, or $\rho$ is cyclic and has two positions $0 \leq i, j$ s.t. $\rho(i) = \rho(j) = s$. Then there exists a prefix $\rho'$ of $\rho$ s.t. $(\rho' = s_0 \rightarrow^* s \rightarrow^+ s)$. Thus we can decide if $\varphi$ holds in $K$ by brute force enumeration of the finite prefixes of the form $(\rho' = s_0 \rightarrow^* s \rightarrow^+ s) \in \Pi_K^{fin}$ where the prefix $(s_0 \rightarrow s)$ is acyclic, and if a run $\rho$ is cyclic and does not satisfy $\psi$ in the prefix $\rho'$ we terminate and answer $K, s_0 \nvDash_0 \varphi$.

We can now conclude that ReachWCTL$^u$ on a finite $n$-WKS is decidable. □

## 3.2 $n$-Weighted Game

We now consider a game with two players and show how synthesis relates to model checking. We define the graph on which the game is played as the game graph.

**Definition 3.1 ($n$-Weighted Game Graph)**
An $n$-Weighted Game Graph ($n$-WGG) is a tuple $G = (S, s_0, \mathcal{AP}, L, T_c, T_u)$ 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 structure of the game graph is an $n$-WKS and we denote the specific $n$-WKS as $K_G$ for the game graph $G$.

**Definition 3.2 ($n$-WKS of a game graph)**
Let $G = (S, s_0, \mathcal{AP}, L, T_c, T_u)$ be a $n$-WGG, then we define $K_G = (S, s_0, \mathcal{AP}, L, T_c \cup T_u)$ as the $n$-WKS constructed from the game graph.

The set of transitions $T_c$ is owned by the controller, and the set $T_u$ is owned by the environment. We write,

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

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 $s$.

Figure 3.2 shows an example of an $n$-WGG, which models a self driving car. In the graph, transitions considered to be uncontrollable are marked as dashed lines, whereas transitions considered controllable are solid lines.

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

**Definition 3.3 (Game)**
Given a $n$-WGG $G$ and a WCTL winning condition $\varphi$ we define the resulting game as the tuple $(G, \varphi)$.

We say that a game defined with a ReachWCTL$^u$ winning condition is a reachability game.

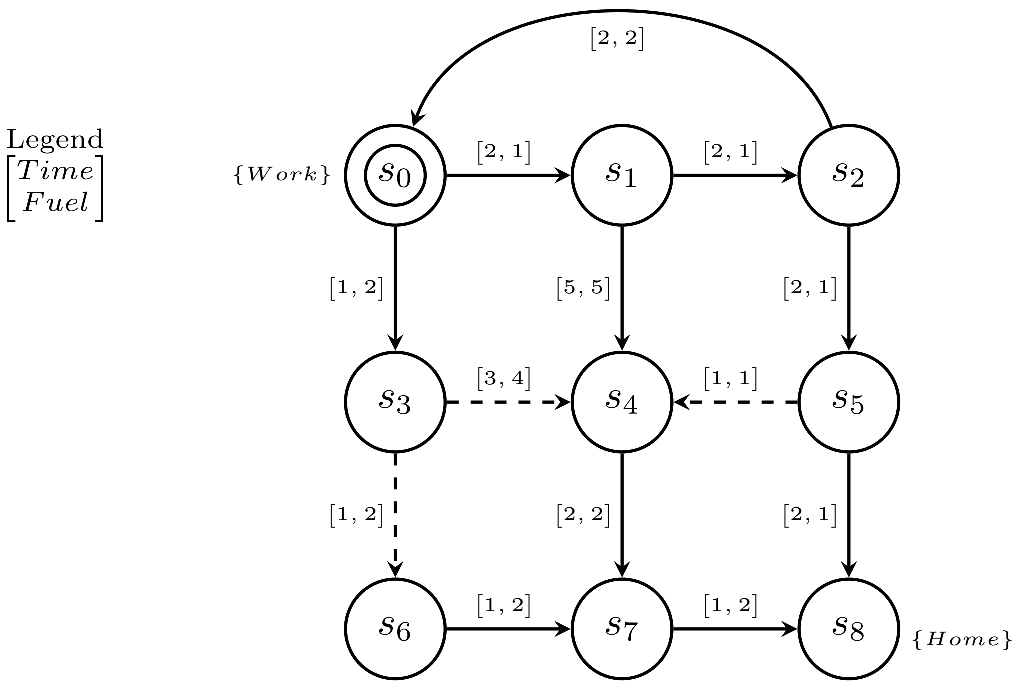


**Figure 3.2:** Example of a 2-WGG where #1 is time spent and #2 is fuel consumption. Arrows indicate controllable transitions and dashed arrows are uncontrollable.

## 3.3 Strategy

In the game a strategy defines a players actions, that is which transitions 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 are only concerned with the controllers strategy. Since we model branching logic, we keep all options of the environment to investigate the branching structure of the Kripke structure resulting from the strategy. We now introduce four different strategy types, and provide results on their expressiveness.

The four strategy types utilizes memory to a different degree. We strive to find the strategy type that utilizes the least amount of memory for a given problem. First we define the *Full Memory Strategy (FM strategy)*, where each move in the game graph is recorded in memory, and future decisions are based thereon. Recall that $\Pi^{fin}_{K_G}$ is the set of all finite runs in $K_G$, starting from $s_0$.

**Definition 3.4 (FM strategy)**
Let $G = (S, s_0, \mathcal{AP}, L, T_c, T_u)$ be an $n$-WGG, then the *Full Memory Strategy (FM strategy)* of the controller is a function $\sigma$ mapping a finite run $\rho$ to a transition going from $\textsc{Last}(\rho)$, where

$$\sigma : \Pi^{fin}_{K_G} \to T_c \cup \{\textsc{nil}\},$$

and $\textsc{nil}$ is the choice to do nothing. We restrict the use of $\textsc{nil}$ s.t. for any $\rho \in \Pi^{fin}_{K_G}$ we have $\sigma(\rho) = \textsc{nil}$ only if $\textsc{Last}(\rho) \not\to$ and if $\sigma(\rho) = s \xrightarrow{\overline{w}} s'$ then $s = \textsc{Last}(\rho)$.

Second, we define the *All States Cost Strategy (ASC strategy)*, where instead of recording each move in memory, only the states visited are recorded, along with the cost of the run.

**Definition 3.5 (ASC strategy)**
A strategy $\sigma$ is an *All States Cost Strategy (ASC strategy)* if for all $\rho, \rho' \in \Pi^{fin}_{K_G}$ we have $\sigma(\rho) = \sigma(\rho')$, whenever $\rho(i) = \rho'(i)$ for all $0 \leq i$ and $\textsc{Cost}(\rho) = \textsc{Cost}(\rho')$.

Third, we define the *SSC strategy*, where only the current state is known, along with the cost of thr run.

**Definition 3.6 (SSC strategy)**
A strategy $\sigma$ is a *Single-State Cost Strategy (SSC strategy)* if for all $\rho, \rho' \in \Pi_{K_G}^{fin}$ we have $\sigma(\rho) = \sigma(\rho')$, whenever $\textsc{Last}(\rho) = \textsc{Last}(\rho')$ and $\textsc{Cost}(\rho) = \textsc{Cost}(\rho')$.

The last type of strategy is the *Memoryless Strategy (ML strategy)*. If the controller has an ML strategy then the choice of which transition to choose is only dependent on the current state.

**Definition 3.7 (ML strategy)**
A strategy $\sigma$ is *Memoryless Strategy (ML strategy)* if for all $\rho, \rho' \in \Pi_{K_G}^{fin}$ we have $\sigma(\rho) = \sigma(\rho')$ whenever $\textsc{Last}(\rho) = \textsc{Last}(\rho')$.

Given a strategy $\sigma$, we can apply that strategy onto the $n$-WKS $K_G$ and get a modified $n$-WKS, by only keeping the controllable transitions that are in the co-domain of $\sigma$.

**Definition 3.8 (Strategy restricted $n$-WKS)**
Given a game graph $G = (S, s_0, \mathcal{AP}, L, T_c, T_u)$ and a FM strategy $\sigma$, we define $G{\restriction}\sigma = (S', s_0, \mathcal{AP}, L', T'_c{\restriction}\sigma \cup T'_u)$ as the $n$-WKS resulting from restricting the game graph under the strategy $\sigma$.

- $S' \subseteq \Pi_{K_G}^{fin}$
- $L'(\rho) = L(\textsc{Last}(\rho))$
- $T'_c{\restriction}\sigma = \{(\rho, \overline{w}, (\rho \circ \sigma(\rho))) \mid \rho \in \Pi_{K_G}^{fin}\}$
- $T'_u = \{(\rho, \overline{w}, \rho \circ (\textsc{Last}(\rho) \xrightarrow{\overline{w}} s')) \mid (\textsc{Last}(\rho) \overset{\overline{w}}{\dashrightarrow} s') \in T_u\}$.

When for all $(\rho = \rho_0 \to \rho_1 \to \dots) \in \Pi_{G{\restriction}\sigma}$, it holds for all $i \in \mathbb{N}_0$ that $\rho_i \neq \rho_{ur} \in S'$, then we call $\rho_{ur}$ unreachable. When $\rho_{ur}$ is unreachable, we write that $\textsc{Cost}(\rho_{ur}) = \infty^n$. In future illustrations we only include the part of $G{\restriction}\sigma$ reachable from $s_0$.

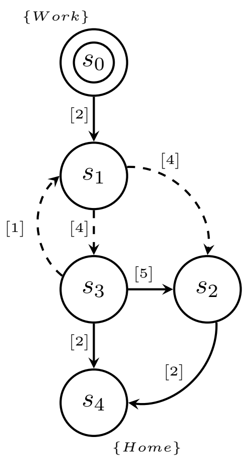
We say that a strategy is a *winning strategy* if the $n$-WKS resulting from restricting the game graph by the strategy satisfy the winning condition.

**Definition 3.9 (Winning Strategy)**
The strategy $\sigma$ is a *winning strategy* over the game $(G, \varphi)$ iff $G\restriction\sigma, s_0 \models_{0^n} \varphi$, where $G = (S, s_0, \mathcal{AP}, L, T_c, T_u)$ and $\varphi$ is a WCTL formula.

Consider the game reachability $(G, AF_{\leq 6}(\{Home\}))$ where $G$ is the 1-WGG illustrated in Figure 3.3a. Given the ML strategy strategy $\sigma$ defined,

$$\sigma(\rho) = \begin{cases} \textsc{nil} & \text{if } \textsc{Last}(\rho) \not\to\text{, else} \\ (s_0, 2, s_1) \in T_c & \text{if } \textsc{Last}(\rho) = s_0\text{, else} \\ (s_2, 2, s_4) \in T_c & \text{if } \textsc{Last}(\rho) = s_2\text{, else} \\ (s_3, 2, s_4) \in T_c & \text{if } \textsc{Last}(\rho) = s_3 \end{cases}$$

we can now illustrate the restricted 1-WKS, $G\restriction\sigma$ in Figure 3.3b.



**(a)** 1-WGG $G$

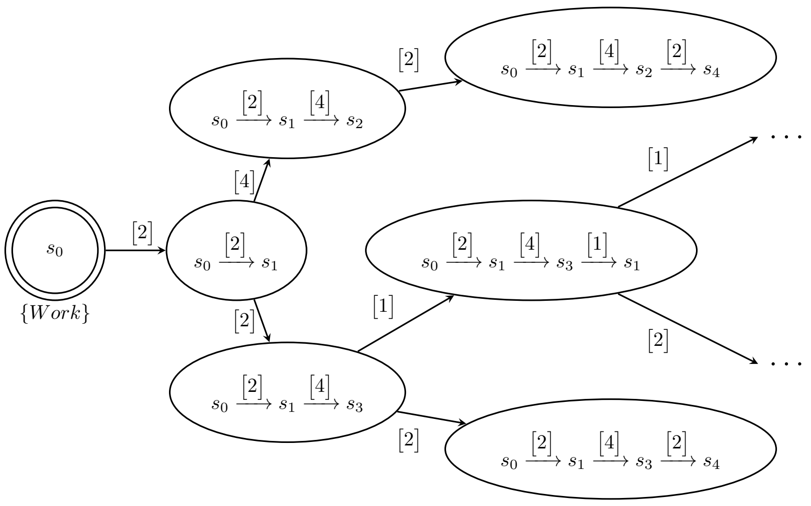

$s_0$
{Work}
[2]
$s_0 \xrightarrow{[2]} s_1$
[4]
$s_0 \xrightarrow{[2]} s_1 \xrightarrow{[4]} s_2$
[2]
$s_0 \xrightarrow{[2]} s_1 \xrightarrow{[4]} s_2 \xrightarrow{[2]} s_4$
[2]
$s_0 \xrightarrow{[2]} s_1 \xrightarrow{[4]} s_3$
[1]
$s_0 \xrightarrow{[2]} s_1 \xrightarrow{[4]} s_3 \xrightarrow{[1]} s_1$
[1]
...
[2]
...
[2]
$s_0 \xrightarrow{[2]} s_1 \xrightarrow{[4]} s_3 \xrightarrow{[2]} s_4$
**(b)** Restricted 1-WKS $G\restriction\sigma$ .

**Figure 3.3:** Example of a strategy and the resulting restricted 1-WKS

Notice that for the reachability game $(G, AF_{\leq 6}(\{Home\}))$ there is no winning strategy as the uncontrollable transition $s_3 \overset{[1]}{\dashrightarrow} s_1$ creates an infinite branch in $G\restriction\sigma$ that does not satisfy $\psi$ within the upper-bound.

### 3.3.1 Strategy Expressiveness

The expressiveness of the strategies is illustrated in Figure 3.4 where ML strategy is the least expressive and FM strategy is the most expressive. Each type of strategy has a subset of the logic for which it is the smallest strategy required. This is ascertained from propositions 1 to 3.

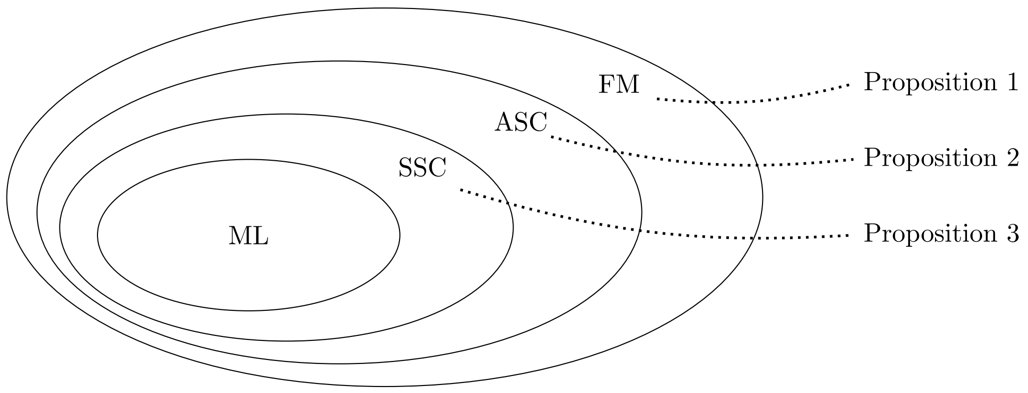


**Figure 3.4:** Classification of the strategies over $n$-WGs.

**Proposition 1**
There is a game $(G, \varphi)$ with a winning FM strategy, but no winning ASC strategy.

**Proof.** Let $(G, \varphi)$ be a game, where $G$ is the game graph illustrated in Figure 3.5 and the WCTL formula $\varphi$ is shown below.

$$\varphi = AF \begin{pmatrix} (\alpha \wedge \#1 \leq 1 \wedge AF(\#1 = 4 \wedge \beta)) \vee \\ (\alpha \wedge \#1 \geq 2 \wedge AF(\#1 = 4 \wedge \gamma)) \end{pmatrix}, \text{ where } \alpha, \beta, \gamma \in \mathcal{AP}$$

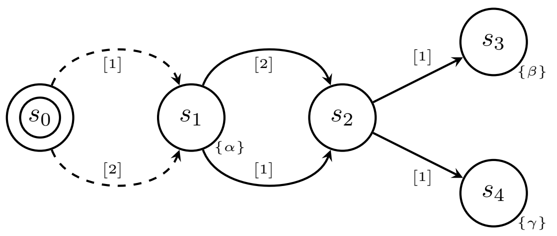


**Figure 3.5:** Game graph $G$, where an ASC strategy is insufficient for WCTL logic.

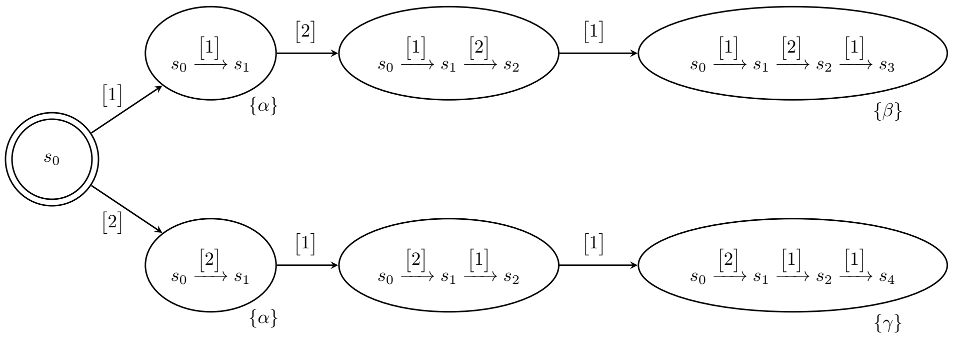
Now, there exists a winning FM strategy $\sigma_1$ for $(G, \varphi)$ and it is defined as:

$\sigma_1(s_0) = \textsc{nil}$

$\sigma_1(s_0 \xrightarrow{1} s_1) = s_1 \xrightarrow{2} s_2$ $\quad$ $\sigma_1(s_0 \xrightarrow{2} s_1) = s_1 \xrightarrow{1} s_2$

$\sigma_1(s_0 \xrightarrow{1} s_1 \xrightarrow{2} s_2) = s_2 \xrightarrow{1} s_3$ $\quad$ $\sigma_1(s_0 \xrightarrow{2} s_1 \xrightarrow{1} s_2) = s_2 \xrightarrow{1} s_4$

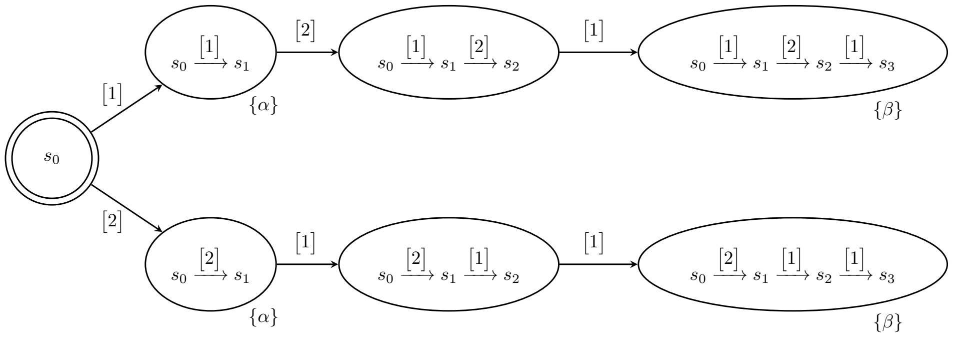
We can construct $G\restriction\sigma_1 = (S', s_0, \mathcal{AP}, L', T'_c\restriction\sigma_1 \cup T'_u)$ s.t. $G\restriction\sigma_1, s_0 \vDash_{0^n} \varphi$, illustrated in Figure. 3.6.



[1]
[2]
$s_0$
$s_0 \xrightarrow{[1]} s_1$
$\{\alpha\}$
$s_0 \xrightarrow{[1]} s_1 \xrightarrow{[2]} s_2$
$s_0 \xrightarrow{[1]} s_1 \xrightarrow{[2]} s_2 \xrightarrow{[1]} s_3$
$\{\beta\}$
$s_0 \xrightarrow{[2]} s_1$
$\{\alpha\}$
$s_0 \xrightarrow{[2]} s_1 \xrightarrow{[1]} s_2$
$s_0 \xrightarrow{[2]} s_1 \xrightarrow{[1]} s_2 \xrightarrow{[1]} s_4$
$\{\gamma\}$

**Figure 3.6:** $G\restriction\sigma_1$ s.t. $G\restriction\sigma, s_0 \vDash_{0^n} \varphi$

We will now show that there is no ASC strategy for $(G, \varphi)$ that is winning. Assume $\sigma_2$ is a winning ASC strategy for $(G, \varphi)$. We get, from Definition 3.5, that $\sigma_2(s_0 \xrightarrow{1} s_1 \xrightarrow{2} s_2) = \sigma_2(s_0 \xrightarrow{2} s_1 \xrightarrow{1} s_2)$, we call this transition $t$. Therefore $G\restriction\sigma_2$ will either be as illustrated in Figure 3.7, where the only label at the tree leaves are $\beta$, or the opposite where only leaves with $\gamma$ exists. This means that we cannot construct $\sigma_2$ s.t. $\beta$ is a label in the leaf, when the environment has chosen $s_0 \overset{1}{\dashrightarrow} s_1$ from the initial state, while $\gamma$ is a label in the leaf, when the environment has chosen $s_0 \overset{2}{\dashrightarrow} s_1$.



[1]
[2]
$s_0$
$s_0 \xrightarrow{[1]} s_1$
$\{\alpha\}$
$s_0 \xrightarrow{[1]} s_1 \xrightarrow{[2]} s_2$
$s_0 \xrightarrow{[1]} s_1 \xrightarrow{[2]} s_2 \xrightarrow{[1]} s_3$
$\{\beta\}$
$s_0 \xrightarrow{[2]} s_1$
$\{\alpha\}$
$s_0 \xrightarrow{[2]} s_1 \xrightarrow{[1]} s_2$
$s_0 \xrightarrow{[2]} s_1 \xrightarrow{[1]} s_2 \xrightarrow{[1]} s_3$
$\{\beta\}$

**Figure 3.7:** $G\restriction\sigma_2$ s.t. $G\restriction\sigma_2, s_0 \nvDash_{0^n} \varphi$

Hence there exists a winning FM strategy, but there does not exist a winning ASC strategy for $(G, \varphi)$. □

**Proposition 2**
There is a game $(G, \varphi)$ with a winning ASC strategy, but no winning SSC strategy.

**Proof.** Let $(G, \varphi)$ be a game, where $G$ is the game graph illustrated in Figure 3.8 and the WCTL formula $\varphi$ is shown below.

$$\varphi = AF \begin{pmatrix} (\alpha \wedge AF(\delta)) \vee \\ (\beta \wedge AF(\gamma)) \end{pmatrix} \qquad \text{where } \alpha, \beta, \gamma, \delta \in \mathcal{AP}$$

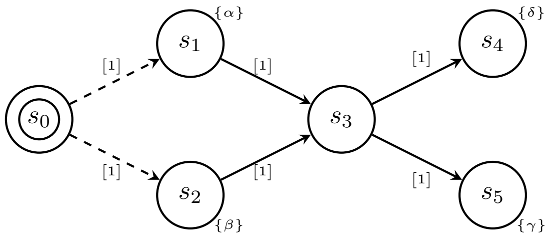


**Figure 3.8:** Game graph $G$, where a SSC strategy is insufficient for WCTL logic.

Now, there exist a winning ASC strategy $\sigma_1$ for $(G, \varphi)$ and it is defined as:

$\sigma_1(s_0) = \textbf{NIL}$

$\sigma_1(s_0 \xrightarrow{[1]} s_1) = s_1 \xrightarrow{[1]} s_3$ $\qquad$ $\sigma_1(s_0 \xrightarrow{[1]} s_2) = s_2 \xrightarrow{[1]} s_3$

$\sigma_1(s_0 \xrightarrow{[1]} s_1 \xrightarrow{[1]} s_3) = s_3 \xrightarrow{[1]} s_4$ $\qquad$ $\sigma_1(s_0 \xrightarrow{[1]} s_2 \xrightarrow{[1]} s_3) = s_3 \xrightarrow{[1]} s_5$

We can construct $G\restriction\sigma_1 = (S', s_0, \mathcal{AP}, L', T'_c\restriction\sigma_1 \cup T'_u)$ s.t. $G\restriction\sigma_1, s_0 \models_{0^n} \varphi$, illustrated in Figure 3.9.

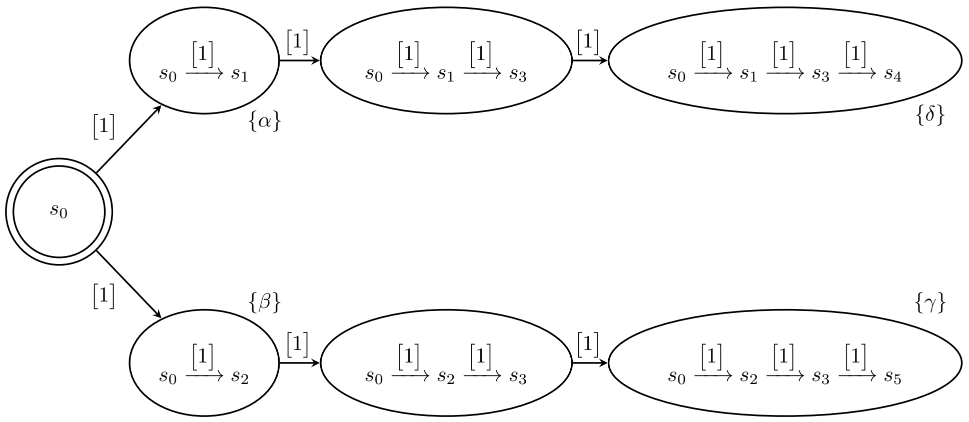
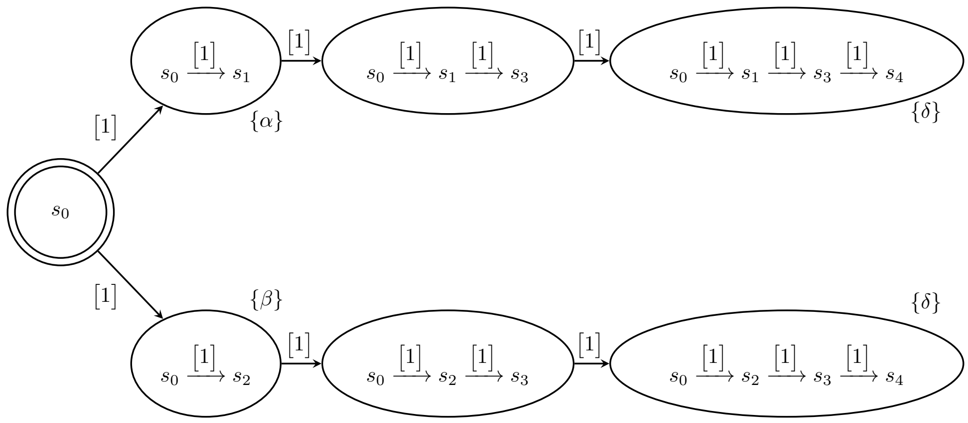

$s_0 \xrightarrow{[1]} s_1$
[1]
$s_0 \xrightarrow{[1]} s_1 \xrightarrow{[1]} s_3$
[1]
$s_0 \xrightarrow{[1]} s_1 \xrightarrow{[1]} s_3 \xrightarrow{[1]} s_4$
[1]
{α}
{δ}
s0
[1]
{β}
{γ}
$s_0 \xrightarrow{[1]} s_2$
[1]
$s_0 \xrightarrow{[1]} s_2 \xrightarrow{[1]} s_3$
[1]
$s_0 \xrightarrow{[1]} s_2 \xrightarrow{[1]} s_3 \xrightarrow{[1]} s_5$

**Figure 3.9:** $G\restriction\sigma_1$, s.t. $G\restriction\sigma, s_0 \models_{0^n} \varphi$

The strategy $\sigma_1$ is winning, because the transition chosen in $s_3$ is dependent on what states were visited previously, and as a SSC strategy cannot differentiate, only one transition is available from $s_3$.



$s_0$
$[1]$
$s_0 \xrightarrow{[1]} s_1$
$\{\alpha\}$
$s_0 \xrightarrow{[1]} s_1 \xrightarrow{[1]} s_3$
$s_0 \xrightarrow{[1]} s_1 \xrightarrow{[1]} s_3 \xrightarrow{[1]} s_4$
$\{\delta\}$
$s_0 \xrightarrow{[1]} s_2$
$\{\beta\}$
$s_0 \xrightarrow{[1]} s_2 \xrightarrow{[1]} s_3$
$s_0 \xrightarrow{[1]} s_2 \xrightarrow{[1]} s_3 \xrightarrow{[1]} s_4$
$\{\delta\}$

**Figure 3.10:** $G\restriction\sigma_2$, s.t. $G\restriction\sigma_2, s_0 \nvDash_{0^n} \varphi$

This means that when constructing the SSC strategy $\sigma_2$, then $\sigma_2(s_0 \xrightarrow{[1]} s_1 \xrightarrow{[1]} s_3) = \sigma_2(s_0 \xrightarrow{[1]} s_2 \xrightarrow{[1]} s_3)$ as illustrated in Figure 3.10, where $\sigma_2(s_0 \xrightarrow{[1]} s_1 \xrightarrow{[1]} s_3) = \sigma_2(s_0 \xrightarrow{[1]} s_2 \xrightarrow{[1]} s_3) = s_3 \xrightarrow{[1]} s_4$ and only $\delta$ is available in the leaves. Otherwise if $\sigma_2(s_0 \xrightarrow{[1]} s_1 \xrightarrow{[1]} s_3) = \sigma_2(s_0 \xrightarrow{[1]} s_2 \xrightarrow{[1]} s_3) = s_3 \xrightarrow{[1]} s_5$ then only $\beta$ is available in the leaves.

Hence there exists a winning ASC strategy, but there does not exist a winning SSC strategy for $(G, \varphi)$. □

**Proposition 3**
There is a reachability game $(G, \varphi)$, with a winning SSC strategy, but no winning ML strategy.

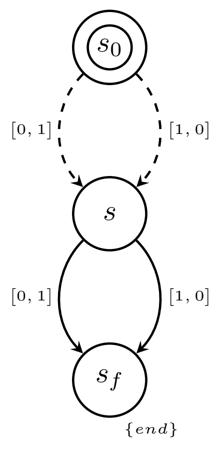
**Proof.** Let $(G, AF_{\leq[1,1]}(end))$ be a reachability game, where $G$ is illustrated in Figure 3.11a. Let $\sigma_1$ be a winning SSC strategy defined as:

$$\sigma_1(s_0) = \textsc{nil} \qquad \sigma_1(s_0 \xrightarrow{[1,0]} s) = s \xrightarrow{[0,1]} s_f \qquad \sigma_1(s_0 \xrightarrow{[0,1]} s) = s \xrightarrow{[1,0]} s_f$$

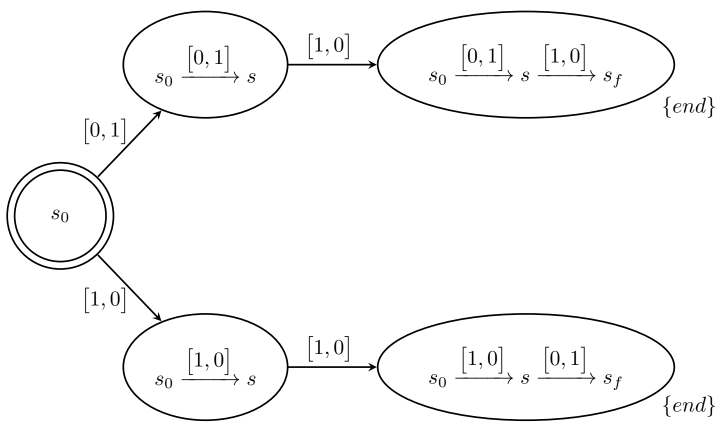
This is illustrated in Figure 3.11b.

We will now show that there exists no ML strategy for $(G, AF_{\leq[1,1]}end)$. Assume there exists a winning ML strategy $\sigma_2$. By Definition 3.7 we get that $\sigma_2(s_0 \xrightarrow{[1,0]} s) = \sigma_2(s_0 \xrightarrow{[0,1]} s)$ and let that transition be $t$. Now, There are two cases:

1. Either $t = s \xrightarrow{[1,0]} s_f$, and $\sigma_2$ cannot be winning, as the branch $s_0 \xrightarrow{[1,0]} s \xrightarrow{[1,0]} s_f$ exceeds the upper-bound of $[1,1]$.



**(a)** Game graph $G$


$s_0$
$[0,1]$
$s_0 \xrightarrow{[0,1]} s$
$[1,0]$
$s_0 \xrightarrow{[0,1]} s \xrightarrow{[1,0]} s_f$
$\{end\}$
$[1,0]$
$s_0 \xrightarrow{[1,0]} s$
$[1,0]$
$s_0 \xrightarrow{[1,0]} s \xrightarrow{[0,1]} s_f$
$\{end\}$

**(b)** $G{\restriction}\sigma_1$, s.t. $G{\restriction}\sigma, s_0 \models_{0^n} AF_{\leq[1,1]}(end)$

**Figure 3.11:** Example of a reachability game where there exist a winning SSC strategy but not a winning ML strategy.

2. Or $t = s \xrightarrow{[0,1]} s_f$ and $\sigma_2$ cannot be winning, as the branch $s_0 \xrightarrow{[0,1]} s \xrightarrow{[0,1]} s_f$ exceeds the upper-bound of $[1,1]$.

Hence there exist a winning strategy SSC strategy for $(G, AF_{\leq[1,1]}(end))$ but not a winning ML strategy. $\square$

We have proven that each strategy has a unique sub-logic for which it is the smallest sufficient strategy guaranteed to be winning.

### 3.3.2 Strategy for $n$-Weighted Games

In the remainder of this paper we focus on the ReachWCTL$^u$ subset of the WCTL. We find that in relation to synthesis the ReachWCTL$^u$ formula $\varphi = EF_{\leq\bar{c}}\psi$ is not very interesting as it reduces to model checking.

**Proposition 4**
Let $(G, EF_{\leq\bar{c}}\psi)$ be a game where $G = (S, s_0, \mathcal{AP}, L, T_c, T_u)$ is an $n$-WGG. Then the synthesis problem for $(G, EF_{\leq\bar{c}}\psi)$ has a solution iff $K_G, s_0 \models_{0^n} EF_{\leq\bar{c}}\psi$.

**Proof.** Let $(G, EF_{\leq\bar{c}}\psi)$ be a game where $G = (S, s_0, \mathcal{AP}, L, T_c, T_u)$ is an $n$-WGG.

$\Rightarrow$ Let $\sigma$ be winning for $(G, EF_{\leq\bar{c}}\psi)$. Since $G{\restriction}\sigma$ is a restriction of $K_G$ and $G{\restriction}\sigma, s_0 \models_{0^n} EF_{\leq\bar{c}}\psi$, then trivially $K_G, s_0 \models_{0^n} EF_{\leq\bar{c}}\psi$.

$\Leftarrow$ Let $K_G, s_0 \models_{0^n} EF_{\leq\bar{c}}\psi$. By the semantics, there must exist a run $\rho_{witness} \in \Pi^{Max}_{K_G}$ where there exists an $0 \leq i$ s.t. $\rho_{witness}(i) \models \psi$. Let $\Pi_{\rho_{witness}}$ be the set of prefixes of $\rho_{witness}$. The solution to the synthesis problem for $(G, EF_{\leq\bar{c}}\psi)$, is then to construct a strategy $\sigma$ s.t. the run $\rho' \in \Pi_{\rho_{witness}}$ is a reachable state in $G\restriction\sigma$, where $\textsc{Last}(\rho') = \rho_{witness}(i)$. We then define $\sigma(\rho)$ for all $\rho \in \Pi^{fin}_{K_G}$ as:

$$\sigma(\rho) = \begin{cases} \textsc{Last}(\rho) \xrightarrow{\overline{w}} s & \text{If there exists } (\rho \circ (\textsc{Last}(\rho) \xrightarrow{\overline{w}} s)) \in \Pi_{\rho_{witness}} \\ & \text{ and } (\textsc{Last}(\rho) \xrightarrow{\overline{w}} s) \in T_c \\ \textsc{Last}(\rho) \xrightarrow{\overline{w}} s' & \text{else if there exists } \textsc{Last}(\rho) \xrightarrow{\overline{w}} s' \in T_c, \\ \textsc{nil} & \text{otherwise.} \end{cases}$$

As $\sigma$ does not restrict uncontrollable transitions, these are trivially preserved. Hence $\rho'$ is a reachable state in $G\restriction\sigma$ and $G\restriction\sigma, s_0 \models_{0^n} EF_{\leq\bar{c}}\psi$, thus $\sigma$ is winning for $(G, EF_{\leq\bar{c}}\psi)$.

□

**Remark 2**
By Proposition 4 an *EF* reachability objective in synthesis reduces to the model checking problem. In games with two players we want to ensure that whatever the environment does, the controller must always be able to reach a state satisfying the proposition. Thus we are interested in all runs in the underlying $n$-WKS and they should all satisfy the proposition, for the controller to have a *winning strategy*. Therefore, from now on, we only consider the following subset of ReachWCTL$^u$, when exploring reachability objectives.

$$\begin{aligned} \varphi &:= AF_{\leq\bar{c}}\psi \\ \psi &:= a \mid \psi_1 \vee \psi_2 \mid \psi_1 \wedge \psi_2 \end{aligned}$$

We will now prove that a reachability game $(G, AF_{\leq\bar{c}}\psi)$ can be solved using a SSC strategy. First, we define the the following sets of prefixes of an $n$-WKS.

**Definition 3.10 (Potentially Satisfying Prefixes)**
Let $(G, AF_{\leq\bar{c}}\psi)$ be a reachability game and $\sigma$ be a winning strategy for that game. We then define the set of *Potentially Satisfying Prefixes* of $G\restriction\sigma$ as:

$$\Pi^{PSP}_{G\restriction\sigma} = \{\rho_i \mid \rho = \rho_0 \to \rho_1 \to \rho_2 \to \cdots \to \rho_k \in \Pi^{fin}_{G\restriction\sigma} \text{ and } \rho_i \not\models \psi \text{ for all } 0 \leq i \leq k\}$$

Notice, that for $G\restriction\sigma = (S', s_0, \mathcal{AP}, L', T'_c\restriction\sigma \cup T'_u)$ we have that $\Pi^{PSP}_{G\restriction\sigma} \subseteq S'$.

**Definition 3.11 (Distance function)**
Let $K = (S, s_0, \mathcal{AP}, L, T)$ be a finite $n$-WKS s.t $s_0 \models \varphi$ where $\varphi = AF_{\leq \bar{c}}\psi$. For all states $s \in S$ we define the distance to a state $s' \in S$ s.t. $s' \models \psi$ as a function $Dist_{(K,\varphi)} : S \to \mathbb{N}_\infty$, where

$$Dist_{(K,\varphi)}(s) = max\{n \mid s \to^n s' \text{ and } s' \models \psi\}$$

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

**Remark 3**
Recall that given the $n$-WKS $G{\restriction}\sigma = (S', s_0, \mathcal{AP}, L', T'_c{\restriction}\sigma \cup T'_u)$ we have that a state in $S'$ is a finite run $\rho \in \Pi^{fin}_{K_G}$.

Observe that given the $n$-WKS $G{\restriction}\sigma = (S', s_0, \mathcal{AP}, L', T'_c{\restriction}\sigma \cup T'_u)$ where $\sigma$ is a winning strategy for a reachability game $(G, AF_{\leq \bar{c}}\psi)$ we have that $Dist(\rho)$ is finite for all $\rho \in \Pi^{PSP}_{G{\restriction}\sigma}$.

We show that when a game $(G, AF_{\leq \bar{c}}\psi)$ has a winning FM strategy $\sigma_1$, there exists a winning SSC strategy $\sigma_2$. First we define that whenever two runs have that same cost and end in the same state they are SSC equivalent.

**Definition 3.12 (SSC equivalent run)**
We define that two finite runs $\rho, \rho' \in \Pi^{fin}_{K_G}$ are SSC equivalent if and only if

$$\textsc{Last}(\rho) = \textsc{Last}(\rho') \land \textsc{Cost}(\rho) = \textsc{Cost}(\rho')$$

We write that $\rho \overset{\text{SSC}}{\equiv} \rho'$, when $\rho$ and $\rho'$ are SSC equivalent.

We then define the set of states in $G{\restriction}\sigma$ which are SSC equivalent to some run $\rho \in \Pi^{fin}_{K_G}$ as the set of SSC equivalent prefixes.

**Definition 3.13 (SSC equivalent prefixes)**
Let $(G, AF_{\leq \bar{c}}\psi)$ be a reachability game where $G = (S, s_0, \mathcal{AP}, L, T_c, T_u)$. Assume there exists an FM strategy $\sigma$ s.t. $G{\restriction}\sigma, s_0 \models AF_{\leq \bar{c}}\psi$. We define SSC equivalent prefixes as a function $\Pi^{\overset{\text{SSC}}{\equiv}} : \Pi^{fin}_{K_G} \to \mathcal{P}(\Pi^{PSP}_{G{\restriction}\sigma})$ s.t.

$$\Pi^{\overset{\text{SSC}}{\equiv}}_{G{\restriction}\sigma}(\rho) = \{\rho' \mid \rho' \in \Pi^{PSP}_{G{\restriction}\sigma} \land \rho \overset{\text{SSC}}{\equiv} \rho'\}$$

We now define how to construct a winning SSC strategy $\sigma_2$ from the restricted $n$-WKS $G\restriction\sigma_1$ for some winning FM strategy $\sigma_1$. We then show that $\sigma_2$ is constructed s.t. $G\restriction\sigma_2, s_0 \models AF_{\leq \bar{c}}\psi$.

**Definition 3.14 (Constructed SSC strategy)**
Let $(G, AF_{\leq \bar{c}}\psi)$ be a reachability game where $G = (S, s_0, \mathcal{AP}, L, T_c, T_u)$. Assume there exists an FM strategy $\sigma_1$ s.t. $G\restriction\sigma_1, s_0 \models AF_{\leq \bar{c}}\psi$. We construct the SSC strategy $\sigma_2$ by doing the following for all $\rho \in \Pi^{fin}_{K_G}$:

$$\sigma_2(\rho) = \begin{cases} \text{NIL} & \text{if } \textsc{Last}(\rho) \not\rightarrow\text{, else} \\ (\textsc{Last}(\rho), \overline{w}, s) \in T_c & \text{if } \Pi^{\overset{\text{SSC}}{\equiv}}_{G\restriction\sigma_1}(\rho) = \emptyset\text{, else} \\ \sigma_1(\rho') & \rho' = \underset{\rho'' \in \Pi^{\overset{\text{SSC}}{\equiv}}_{G\restriction\sigma_1}(\rho)}{\arg\min}\ Dist_{(G\restriction\sigma_1, \varphi)}(\rho'') \end{cases} \tag{3.1}$$

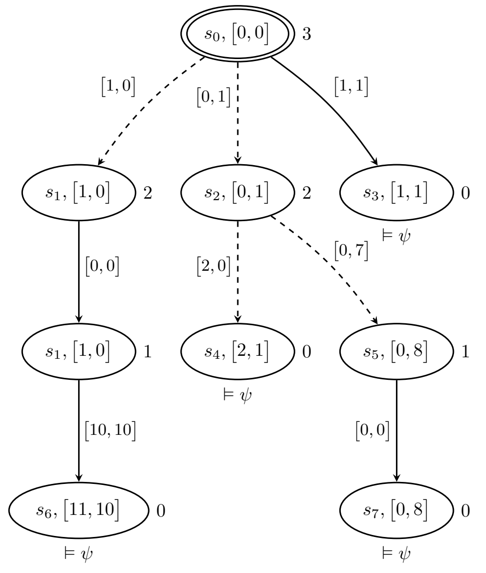
As all SSC equivalent runs $\rho, \rho' \in \Pi^{fin}_{K_G}$ which have an SSC equivalent prefix in $G\restriction\sigma_1$ have exactly the same output we guarantee that $\sigma_2$ is indeed a SSC strategy. For a visual illustration of the construction, referrer to Figure 3.12.

Notice here how there are two states $\rho, \rho' \in S'$, where $\textsc{Last}(\rho) = \textsc{Last}(\rho') = s_1$ and $\textsc{Cost}(\rho) = \textsc{Cost}(\rho') = [1,0]$. When constructing the SSC strategy $\sigma_2$ for these two runs, then $\sigma_2(\rho) = \sigma_2(\rho') = s_1 \xrightarrow{[10,10]} s_6$, since this is the strategy in $\sigma_1$ for the run $\rho$ with the shortest distance. If we had constructed the strategy with the longest distance then we would have that $\sigma_2(\rho) = \sigma_2(\rho') = s_1 \xrightarrow{[0,0]} s_1$ and we would repeat an infinite loop. This strategy will obviously not be winning. Based on this observation we define the measure as the distance from the SSC equivalent prefix and verify that the measure is always decreasing.

**Definition 3.15 (SSC measure)**
Let $(G, \varphi)$ be a reachability game where $G = (S, s_0, \mathcal{AP}, L, T_c, T_u)$ and $\varphi = AF_{\leq \bar{c}}\psi$. Let $\sigma_1$ is a winning FM strategy for that game and we define a measure function $m : \Pi^{fin}_{K_G} \to \mathbb{N}_\infty$ s.t.

$$m^{\overset{\text{SSC}}{\equiv}}_{G\restriction\sigma_1}(\rho) = \min_{\rho' \in \Pi^{\overset{\text{SSC}}{\equiv}}_{G\restriction\sigma_1}(\rho)} \left(Dist_{(G\restriction\sigma_1, \varphi)}(\rho')\right), \tag{3.2}$$

where $\min \emptyset = \infty$.



**(a)**

| LAST$(\rho)$, COST$(\rho)$ | $\sigma_2(\rho)$ |
|---|---|
| $s_0, \begin{bmatrix}0,0\end{bmatrix}$ | $s_0 \xrightarrow{[1,1]} s_3$ |
| $s_1, \begin{bmatrix}1,0\end{bmatrix}$ | $s_1 \xrightarrow{[10,10]} s_6$ |
| $s_2, \begin{bmatrix}0,1\end{bmatrix}$ | NIL |
| $s_3, \begin{bmatrix}1,1\end{bmatrix}$ | NIL |
| $s_4, \begin{bmatrix}2,1\end{bmatrix}$ | NIL |
| $s_5, \begin{bmatrix}0,8\end{bmatrix}$ | $s_5 \xrightarrow{[0,0]} s_7$ |
| $s_6, \begin{bmatrix}11,10\end{bmatrix}$ | NIL |
| $s_7, \begin{bmatrix}0,8\end{bmatrix}$ | NIL |

**(b)**

**Figure 3.12:** Figure 3.12a is an example of a restricted game graph $G{\restriction}\sigma_1 = (S', s_0, \mathcal{AP}, L', T'_c{\restriction}\sigma_1 \cup T'_u)$, where $\sigma_1$ is an FM strategy. The last state of a run $\rho \in S'$ along with the cost of $\rho$ represents the states, and the number outside each state is the distance s.t. it is equal to $Dist_{(G{\restriction}\sigma_1,\varphi)}(\rho)$. In Figure 3.12b the construction of $\sigma_2$ for each run $\rho$ is illustrated in the two columns

Observe from Definition 3.15 that if $Dist_{(G{\restriction}\sigma_1,\varphi)}(s_0)$ is finite, then so is $m^{\overset{\text{SSC}}{\equiv}}_{G{\restriction}\sigma_1}(s_0)$. For simplicity we will omit $G{\restriction}\sigma$ from the notation s.t.

$$\Pi^{PSP} = \Pi^{PSP}_{G{\restriction}\sigma_2} \qquad \Pi^{\overset{\text{SSC}}{\equiv}} = \Pi^{\overset{\text{SSC}}{\equiv}}_{G{\restriction}\sigma_1} \qquad m = m^{\overset{\text{SSC}}{\equiv}}_{G{\restriction}\sigma_1} \qquad Dist = Dist_{(G{\restriction}\sigma_1,\varphi)}$$

**Lemma 3**
Let $(G, AF_{\leq \vec{c}}\psi)$ be a reachability game where $G = (S, s_0, \mathcal{AP}, L, T_c, T_u)$, $\sigma_1$ is a winning FM strategy for that game, and $\sigma_2$ a SSC strategy constructed by Definition 3.14. If there exists $\rho \in \Pi^{PSP}$ s.t. $m(\rho)$ is finite and $\rho' = \rho \circ t$, where $t \in T_u \cup \sigma_2(\rho)$, $t = (\text{LAST}(\rho) \xrightarrow{\overline{w}} s)$ and $s \in S$ then $m(\rho') < m(\rho)$.

**Proof.** Let $m(\rho)$ be finite. Then by Definition 3.15 it holds that $\Pi^{\overset{\text{SSC}}{\equiv}}(\rho) \neq \emptyset$ and then by Definition 3.14 we have that $\rho' = \rho \circ t$, where $t \in T_u \cup \sigma_2(\rho)$ and $t = \text{LAST}(\rho) \xrightarrow{\overline{w}} s$, for some $s \in S$. We show that $m(\rho') < m(\rho)$. Since $m(\rho)$ is finite there exists a $\rho_{min} \in \Pi^{\overset{\text{SSC}}{\equiv}}(\rho)$ s.t. $m(\rho) = Dist(\rho_{min})$. By Definition 3.14 observe that $(\rho_{min} \circ t) \in \Pi^{\overset{\text{SSC}}{\equiv}}(\rho')$, hence

$$m(\rho') \leq Dist(\rho_{min} \circ t) < Dist(\rho_{min}) = m(\rho).$$

Thus $m(\rho') < m(\rho)$. □

**Theorem 4**
If there is a winning FM strategy for a reachability game $(G, AF_{\leq \bar{c}}\psi)$ then there is a winning SSC strategy for $(G, AF_{\leq \bar{c}}\psi)$.

**Proof.** Let $\sigma_1$ be a winning FM strategy for $(G, AF_{\leq \bar{c}}\psi)$, and let $\sigma_2$ be constructed as defined in Definition 3.14. Then by Lemma 3 we know that for all $\rho_{max} \in \Pi^{Max}_{G\restriction\sigma_2}$, there exists a position $0 \leq i$ s.t. either $m(\rho_{max}(i)) > 0$ and the measure is decreasing or $m(\rho_{max}(i)) = 0$ hence $\rho_{max}(i) \vDash \psi$. Observe that $\textsc{Cost}_{\rho_{max}}(i) \leq \bar{c}$ as $\textsc{Cost}_{\rho_{max}}(i) = \textsc{Cost}(\rho)$. □

It follows from theorem 4 that a reachability game $(G, AF_{\leq \bar{c}}\psi)$ has a SSC strategy whenever there exits a winning strategy for that game

## 3.4 Example: Self-driving car

We consider a self driving car, where the car has to choose a route from work to home. A central system may influence the route of the car, based on the current road data, to avoid congestion.

In Figure 3.2, the road network is illustrated in a 2-WGG $G = (S, s_0, \mathcal{AP}, L, T_c, T_u)$. The states are road intersections and transitions depicts the roads between intersections. Each road has an associated weight vector, where the first component is time and the second is the fuel consumed. Roads chosen by the central system are shown in dashed lines, and roads chosen by the car are solid.

Let us say the car has 8 units of fuel, and we would like to reach *Home* without a refuel. To ensure this we need to devise a strategy, such that no matter the environment actions, we will still reach home using no more than 8 units of fuel. We can formulate this formally as: Given the reachability game $(G, \varphi)$ where $\varphi = AF_{\leq[\infty,8]}(Home)$ does there exists a strategy $\sigma$ such that $G\restriction\sigma, s_0 \vDash \varphi$.

For this reachability game $(G, \varphi)$ there is a finite number SSC strategies within the bound but only one produces a resulting $n$-WKS $G\restriction\sigma$ which satisfies $AF_{\leq[\infty,8]}(Home)$. Formally the winning strategy $\sigma$ is defined as follows:

$$\sigma(s_0) = s_0 \xrightarrow{[2,1]} s_1 \qquad \sigma(s_0 \xrightarrow{[2,1]} s_1 \xrightarrow{[2,1]} s_2 \xrightarrow{[2,1]} s_5) = s_5, \xrightarrow{[2,1]} s_8$$

$$\sigma(s_0 \xrightarrow{[2,1]} s_1) = s_1 \xrightarrow{[2,1]} s_2 \qquad \sigma(s_0 \xrightarrow{[2,1]} s_1 \xrightarrow{[2,1]} s_2 \xrightarrow{[2,1]} s_5 \xrightarrow{[1,1]} s_4) = s_4 \xrightarrow{[2,2]} s_7$$

$$\sigma(s_0 \xrightarrow{[2,1]} s_1 \xrightarrow{[2,1]} s_2) = s_2 \xrightarrow{[2,1]} s_5 \qquad \sigma(s_0 \xrightarrow{[2,1]} s_1 \xrightarrow{[2,1]} s_2 \xrightarrow{[2,1]} s_5 \xrightarrow{[1,1]} s_4 \xrightarrow{[2,2]} s_7) = s_7 \xrightarrow{[1,2]} s_8$$

Note that $\sigma$ has defined input/output for two different runs in the game graph $G$ from Figure 3.2. This is due to the uncontrollable transition encountered in $s_5$, where the central system may choose to send the car on a detour. For each route generated by our strategy we get a finite run in $G\restriction\sigma$ , and we are now able to calculate the cost of each run, by adding each vector in the run.

If we instead were to consider time as the limited resource we could construct the formula $\varphi = AF_{\leq[8,\infty]}(\{Home\})$ giving us the game $(G, AF_{\leq[8,\infty]}(Home))$. Now the winning strategy is defined as follows:

$$\sigma(s_0) = s_0 \xrightarrow{[1,2]} s_3 \qquad \sigma(s_0 \xrightarrow{[1,2]} s_3 \xrightarrow{[3,4]} s_4) = s_4 \xrightarrow{[2,2]} s_7$$

$$\sigma(s_0 \xrightarrow{[1,2]} s_3) = \textbf{NIL} \qquad \sigma(s_0 \xrightarrow{[1,2]} s_3 \xrightarrow{[1,2]} s_6 \xrightarrow{[1,2]} s_7) = s_7 \xrightarrow{[1,2]} s_8$$

$$\sigma(s_0 \xrightarrow{[1,2]} s_3 \xrightarrow{[1,2]} s_6) = s_6 \xrightarrow{[1,2]} s_7 \qquad \sigma(s_0 \xrightarrow{[1,2]} s_3 \xrightarrow{[3,4]} s_4 \xrightarrow{[2,2]} s_7) = s_7 \xrightarrow{[1,2]} s_8$$

While this strategy is still guaranteed to reach the label *Home* the fuel consumption using this strategy can be as high as 10 unites. It is easy to see that if the formula had been specified as $\varphi = AF_{\leq[10,10]}(Home)$ both of these strategies would have been winning strategies.

## 3.5 Synthesis

Church defined the synthesis problem in [5], and in later work Pnueli et al. describe a game-theoretic view of controller synthesis[12]. From this we define the synthesis problem as *"Extraction of the winning strategy for the controller in a two-player game."* Formally,

**Definition 3.16 (Synthesis problem for reachability games)**
Given a reachability game $(G, \varphi)$ where $G = (S, s_0, \mathcal{AP}, L, T_c, T_u)$ and $\varphi = AF_{\leq\bar{c}}\psi$ the synthesis problem is to find a strategy $\sigma$ s.t. $G\restriction\sigma \models \varphi$.

The synthesis problem is closely related to the model checking problem. Given a reachability game $(G, AF_{\leq\bar{c}}\psi)$ and a SSC strategy $\sigma$, we can deduce if $\sigma$ is *winning* by verifying that $G\restriction\sigma \models AF_{\leq\bar{c}}\psi$. There are finitely many SSC strategies in a game, since the formula is given a bound $\bar{c}$ and we therefore only consider strategies where the cost $\overline{w} \leq \bar{c}$. As there are finitely many strategies we need to consider we can extract $\sigma$ by enumerating all the strategies of the game. This brute force approach is illustrated in Algorithm 1.

**Algorithm 1** Brute Force Synthesis Algorithm

```
Input: A n-WG $(G, AF_{\leq \bar{c}} \psi)$.
Output: A winning strategy $\sigma$ for the $\varphi = AF_{\leq \bar{c}} \psi$ over the game graph $G$, if such
    a strategy exists.
1: for all SSC strategies $\sigma$ over $G$ do
2:     if $G \restriction \sigma, s_0 \models_{0^n} \varphi$ then return $\sigma$
   return NoSolution
```

Since the model checking problem for reachability is decidable, and there are finitely many strategies to enumerate, it is clear that the synthesis problem for reachability is also decidable. However, Algorithm 1 suffers from a high complexity limiting its applicability. As such we investigate different methods in an attempt to diminish the complexity.

# Chapter 4

# 1-Weighted Games

In this chapter we prove that for 1-weighted reachability games there is a winning ML strategy if there exists a winning strategy. We then present an on-the-fly technique for synthesizing strategies for 1-weighted reachability games. First we introduce the weighted attractor set method, from which a strategy can be extracted. We then reduce the computation of attractor sets to computing a minimum prefixed-point assignment of a Symbolic Dependency Graph (SDG), by presenting an encoding of the game into an SDG. Lastly, we present an algorithm which can extract the strategy from a minimum prefixed-point assignment of an SDG.

In the context of single weighted games, we omit the vector notation, e.g. $\overline{w} \in \mathbb{N}_0^1$ becomes $k \in \mathbb{N}_0$.

## 4.1 Strategy for 1-Weighted Games

We show that when a 1-WG $(G, AF_{\leq c}\psi)$ has a winning SSC strategy $\sigma_1$, then there exists a winning ML strategy $\sigma_2$. First we define that whenever two runs end in the same state they are ML equivalent.

**Definition 4.1 (ML equivalent run)**
We define that two finite runs $\rho, \rho' \in \Pi_{K_G}^{fin}$ are ML equivalent if and only if

$$\textsc{Last}(\rho) = \textsc{Last}(\rho')$$

We write that $\rho \overset{\text{ML}}{\equiv} \rho'$, when $\rho$ and $\rho'$ are ML equivalent.

We then define the set of states in $G{\restriction}\sigma$ which are ML equivalent to some run $\rho \in \Pi_{K_G}^{fin}$ as the set of ML equivalent prefixes.

**Definition 4.2 (ML equivalent prefixes)**
Let $(G, AF_{\leq c}\psi)$ be a reachability game where $G = (S, s_0, \mathcal{AP}, L, T_c, T_u)$ is a 1-WGG and let $\sigma$ be a winning strategy of that game. We define the memoryless equivalent prefixes as a function $\Pi^{\overset{\text{ML}}{\equiv}}_{G\restriction\sigma} : \Pi^{fin}_{K_G} \to \mathcal{P}(\Pi^{PSP}_{G\restriction\sigma})$:

$$\Pi^{\overset{\text{ML}}{\equiv}}_{G\restriction\sigma}(\rho) = \{\rho' \mid \rho' \in \Pi^{PSP}_{G\restriction\sigma} \wedge \rho \overset{\text{ML}}{\equiv} \rho'\}$$

We now define how to construct a winning ML strategy $\sigma_2$ from the restricted 1-WKS $G\restriction\sigma_1$ for some winning SSC strategy $\sigma_1$. We then show that $\sigma_2$ is constructed s.t. $G\restriction\sigma_2, s_0 \models AF_{\leq c}\psi$.

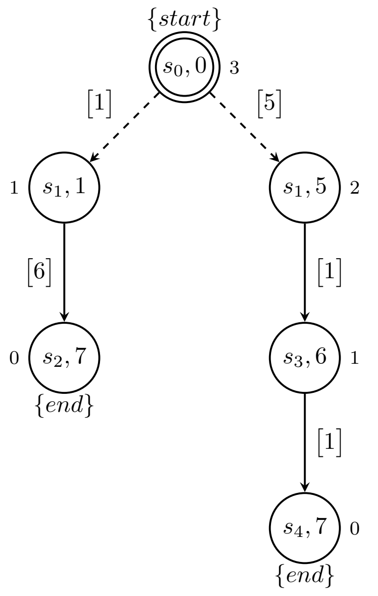
**Definition 4.3 (Constructed ML strategy)**
Let $(G, \varphi)$ be a 1-WG where $G = (S, s_0, \mathcal{AP}, L, T_c, T_u)$ is a 1-WGG and $\varphi = AF_{\leq c}\psi$. Let $\sigma_1$ be a SSC strategy s.t. $G\restriction\sigma_1, s_0 \models \varphi$. We construct the ML strategy $\sigma_2$, in the following way for all $\rho \in \Pi^{fin}_{K_G}$:

$$\sigma_2(\rho) = \begin{cases} \text{NIL} & \text{if } \textsc{Last}(\rho) \not\to, \text{ else} \\ (\textsc{Last}(\rho), \overline{w}, s) \in T_c & \text{if } \Pi^{\overset{\text{ML}}{\equiv}}_{G\restriction\sigma_1}(\rho) = \emptyset, \text{ else} \\ \sigma_1(\rho') & \rho' = \underset{\rho'' \in \Pi^{\overset{\text{ML}}{\equiv}}_{G\restriction\sigma_1}(\rho)}{\arg\max}\ \textsc{Cost}(\rho'') \end{cases} \tag{4.1}$$

As all ML equivalent runs $\rho, \rho' \in \Pi^{fin}_{K_G}$ which have an ML equivalent prefix in $G\restriction\sigma_1$ have exactly the same output we guarantee that $\sigma_2$ is indeed a ML strategy. For a visual illustration of the construction, referrer to Figure 4.1.

Notice here how there are two states $\rho, \rho' \in S'$, where $\textsc{Last}(\rho) = \textsc{Last}(\rho') = s_1$. When constructing the ML strategy $\sigma_2$ for these two runs, we have to account for the cost, as mapping the wrong transition might cause $\sigma_2$ to breach the upper-bound. As such we construct $\sigma_2$ to be cautious, by defining $\sigma_2(\rho) = \sigma_2(\rho') = s_1 \xrightarrow{1} s_3$. If we had constructed the strategy by the shortest distance we would have that $\sigma_2(\rho) = \sigma_2(\rho') = s_1 \xrightarrow{6} s_2$ and this will cause us the breach the upper-bound the restricted game.

As the cost of the ML equivalent runs influence our choices, we also define a measure $m^{\overset{\text{ML}}{\equiv}}_{G\restriction\sigma_1}(\rho)$ for a run $\rho$ s.t. cost influence the measure; in fact, it is the dominant factor.



**(a)** The restricted 1-WKS $G{\restriction}\sigma_1$

| $\textsc{Last}(\rho)$ | $\sigma_2(\rho)$ |
|---|---|
| $s_0$ | NIL |
| $s_1$ | $s_1 \xrightarrow{[1]} s_3$ |
| $s_2$ | NIL |
| $s_3$ | $s_3 \xrightarrow{[1]} s_4$ |
| $s_4$ | NIL |

**(b)** The strategy $\sigma_1$. First column is the domain, and second column is the co-domain.

**Figure 4.1:** Figure 4.1a is an example of a restricted game graph $G{\restriction}\sigma_1 = (S', s_0, \mathcal{AP}, L', T'_c{\restriction}\sigma_1 \cup T'_u)$, where $\sigma_1$ is an SSC strategy. The last state of a run $\rho \in S'$ along with the cost of $\rho$ represents the states, and the number outside each state is the distance s.t. it is equal to $Dist_{(G{\restriction}\sigma_1,\varphi)}(\rho)$. In Figure 4.1b the construction of $\sigma_2$ for each run $\rho$ is illustrated in the two columns.

**Definition 4.4 (ML strategy measure)**
Let $(G, \varphi)$ be a 1-WG where $G = (S, s_0, \mathcal{AP}, L, T_c, T_u)$ is a 1-WGG and $\varphi = AF_{\leq c}\psi$. Let $\sigma_1$ is a winning SSC strategy for that game, and we define a measure function $m^{\overset{ML}{\equiv}}_{G{\restriction}\sigma} : \Pi^{fin}_{K_G} \to \mathbb{N}_\infty$:

$$m^{\overset{ML}{\equiv}}_{G{\restriction}\sigma_1}(\rho) = \begin{cases} \infty & \text{If } \Pi^{\overset{ML}{\equiv}}_{G{\restriction}\sigma_1}(\rho) = \emptyset \\ (c - \max\limits_{\rho' \in \Pi^{\overset{ML}{\equiv}}_{G{\restriction}\sigma_1}(\rho)} \textsc{Cost}(\rho')) * Dist_{(G{\restriction}\sigma_1,\varphi)}(s_0) + Dist_{(G{\restriction}\sigma_1,\varphi)}(\rho') & \text{Otherwise} \end{cases}$$

Observe that if $Dist_{(G{\restriction}\sigma_1,\varphi)}(s_0)$ is finite then $m^{\overset{ML}{\equiv}}_{G{\restriction}\sigma_1}(s_0)$ is finite. For simplicity we will omit $G{\restriction}\sigma$ from the notation s.t.

$$\Pi^{PSP} = \Pi^{PSP}_{G{\restriction}\sigma_2} \qquad \Pi^{\overset{ML}{\equiv}} = \Pi^{\overset{ML}{\equiv}}_{G{\restriction}\sigma_1} \qquad m = m^{\overset{ML}{\equiv}}_{G{\restriction}\sigma_1} \qquad Dist = Dist_{(G{\restriction}\sigma_1,\varphi)}$$

**Lemma 5**
Let $(G, AF_{\leq c}\psi)$ be a 1-WG, where $G = (S, s_0, \mathcal{AP}, L, T_c, T_u)$ is a 1-WGG. Let $\sigma_1$ be a winning SSC strategy for the game, and let $\sigma_2$ be an ML strategy constructed from Definition 4.3. Given $\rho \in \Pi^{PSP}$ then there exists a $\rho_{max} \in \Pi^{\overset{\text{ML}}{\equiv}}(\rho)$ s.t. $\sigma_1(\rho_{max}) = \sigma_2(\rho)$ and we have that $\textsc{Cost}(\rho) \leq \textsc{Cost}(\rho_{max})$.

**Proof.** By induction on the length of $\rho'$. Assume there exist $\rho \in \Pi^{PSP}$ s.t. $\sigma_1(\rho_{max}) = \sigma_2(\rho)$ and $\textsc{Cost}(\rho) \leq \textsc{Cost}(\rho_{max})$, where $\rho_{max} \in \Pi^{\overset{\text{ML}}{\equiv}}(\rho)$. Then by Definition 4.3 $\rho' = \rho \circ t$ where $t \in T_u \cup \sigma_2(\rho)$ and $t = \textsc{Last}(\rho) \xrightarrow{w} s$, for some $s \in S$. Since $\textsc{Cost}(\rho) \leq \textsc{Cost}(\rho_{max})$ we have that $\textsc{Cost}(\rho \circ t) \leq \textsc{Cost}(\rho_{max} \circ t)$ and obviously $(\rho \circ t) \overset{\text{ML}}{\equiv} (\rho_{max} \circ t)$.

Given $s_0 \in \Pi^{PSP}$ it is clear that there exists some run $\rho \in \Pi^{\overset{\text{ML}}{\equiv}}(s_0)$ and as $\textsc{Cost}(s_0) = 0$ then $\textsc{Cost}(s_0) \leq \textsc{Cost}(\rho)$ and thus we have that for all $\rho \in \Pi^{PSP}$ there is a $\rho_{max} \in \Pi^{\overset{\text{ML}}{\equiv}}(\rho)$ s.t. $\sigma_1(\rho_{max}) = \sigma_2(\rho)$ and $\textsc{Cost}(\rho) \leq \textsc{Cost}(\rho_{max})$. □

**Lemma 6**
Let $(G, AF_{\leq c}\psi)$ be a 1-WG where $G = (S, s_0, \mathcal{AP}, L, T_c, T_u)$ is a 1-WGG, $\sigma_1$ is a winning SSC strategy for that game and $\sigma_2$ a strategy constructed by Definition 4.3. If $\rho' = \rho \circ t$ and $t = \textsc{Last}(\rho) \xrightarrow{w} s \in T_u \cup \sigma_2(\rho)$, for some $s \in S$, where $\rho \in \Pi^{PSP}$, then $m(\rho') < m(\rho)$

**Proof.** Let $m(\rho)$ be finite. Then by Definition 4.4 it holds that $\Pi^{\overset{\text{ML}}{\equiv}}(\rho) \neq \emptyset$ and then by Definition 4.3 we have that $\rho' = \rho \circ t$, where $t \in T_u \cup \sigma_2(\rho)$ and $t = \textsc{Last}(\rho) \xrightarrow{w} s$, for some $s \in S$. We show that $m(\rho') < m(\rho)$. Since $m(\rho)$ is finite there exists a $\rho_{max} \in \Pi^{\overset{\text{ML}}{\equiv}}(\rho)$ s.t. $\sigma_1(\rho_{max}) = \sigma_2(\rho)$ and by Definition 4.3 there exists at least one $\rho'_{max} \in \Pi^{\overset{\text{ML}}{\equiv}}(\rho')$ s.t. $\sigma_1(\rho'_{max}) = \sigma_2(\rho')$ and by Lemma 5 $\textsc{Cost}(\rho') \leq \textsc{Cost}(\rho'_{max})$. As the cost of a run can never decrease it follows trivially that $\textsc{Cost}(\rho_{max}) \leq \textsc{Cost}(\rho'_{max})$.

If $\textsc{Cost}(\rho_{max}) = \textsc{Cost}(\rho'_{max})$ then $\textsc{Cost}(\rho_{max} \circ \sigma_1(\rho_{max})) = \textsc{Cost}(\rho'_{max})$ and as $\sigma_1$ is a SSC strategy then it follows that $Dist(\rho'_{max}) = Dist(\rho_{max} \circ \sigma_1(\rho_{max}))$ and thus $Dist(\rho'_{max}) < Dist(\rho_{max})$.

If $\textsc{Cost}(\rho_{max}) < \textsc{Cost}(\rho'_{max})$ then it is trivial to see that as the cost is the dominant factor in $m$ we do not care about distance. Hence, for both cases we have that:

$$\begin{gathered} m(\rho') = (c - \textsc{Cost}(\rho'_{max})) * Dist(s_0) + Dist(\rho'_{max}) \\ < \\ m(\rho) = (c - \textsc{Cost}(\rho_{max})) * Dist(s_0) + Dist(\rho_{max}) \end{gathered}$$

By the observation that if $Dist(s_0)$ is finite, then so is $m(s_0)$, we have that for all $\rho \in \Pi^{PSP}$ and $\rho' = \rho \circ (\textsc{Last}(\rho) \xrightarrow{w} s)$, where $s \in S$ and $m(\rho)$ is finite and that

$m(\rho') < m(\rho)$ whenever $\sigma_1$ is winning for $(G, AF_{\leq c}\psi)$. □

**Theorem 7**
If there is a winning SSC strategy for a 1-WG $(G, AF_{\leq c}\psi)$ then there is a winning ML strategy for $(G, AF_{\leq c}\psi)$.

**Proof.** Let $\sigma_1$ be a winning FM strategy for $(G, AF_{\leq \bar{c}}\psi)$, and let $\sigma_2$ be constructed as defined in Definition 4.3. Then by Lemma 6 we know that for all $\rho_{max} \in \Pi^{Max}_{G\restriction\sigma_2}$ there exists a position $0 \leq i$ s.t. either $m(\rho_{max}(i)) > 0$ and $\rho_{max}(i) \vDash \psi$ or $m(\rho_{max}(i)) = 0$ hence $\rho_{max}(i) \vDash \psi$. By Lemma 5 we get that for all $0 \leq i$ we have that $\textsc{Cost}_{\rho_{max}}(i) \leq c$. □

It now follows from theorem 7 that a 1-weighted reachability game $(G, AF_{\leq c}\psi)$ has a ML strategy whenever there exits a winning strategy for that game.

## 4.2 Attractor Set

One method of synthesizing ML strategy for a 1-weighted reachability game is to extract it from the *attractor set*. Traditionally the attractor set $F_i$ is the set of states from which the controller can force a visit to some final state in $i$ steps or less. Intuitively this gives us a set of states from which the controller has a winning strategy.

Formally, let $(G, AF_{\leq c}\psi)$ be a game, where $G = (S, s_0, \mathcal{AP}, L, T_c, T_u)$ is a 1-WGG. We define $S_f = \{s \in S \mid s \vDash \psi\}$ as the set of final states. We can now define the attractor set $F_i$ as the set of pairs $(s, k)$, where $s \in S$ and $k \in \mathbb{N}_0$, from which the controller can force a visit to a state $s' \in S_f$ in $i$ steps or less with the cost $k \leq c$. The set $F_0$ is simply the pairs consisting of a final state and cost of 0:

$$F_0 = \{(s, 0) \mid s \in S_f\}$$

$F_i$ is then calculated in the following manner for all $i \in \mathbb{N}$.

$$F_i = F_{i-1} \cup Add_i$$

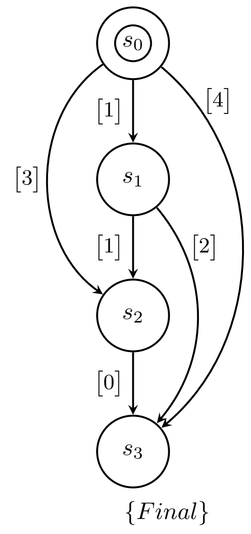
And $Add_i$ is calculated as follows for all $i \in \mathbb{N}$:

$$Add_i = \left\{ (s,k) \;\middle|\; \begin{array}{l} \text{Whenever } s \overset{k_1}{\dashrightarrow} s' \text{ then } \exists (s',k_2) \in F_{i-1} \text{ where } k_1+k_2 \leq c \text{ and} \\ \text{if } s \rightarrow \text{ then } \exists s \xrightarrow{k_1} s' \text{ s.t. } (s',k_2) \in F_{i-1} \text{ where } k_1+k_2 \leq c \text{ and} \\ k = \max \begin{pmatrix} \min\limits_{s \xrightarrow{k_1} s'} \left( \min\limits_{(s',k_2) \in F_{i-1}} (k_1+k_2) \right) \\ \max\limits_{s \overset{k_1}{\dashrightarrow} s'} \left( \min\limits_{(s',k_2) \in F_{i-1}} (k_1+k_2) \right) \end{pmatrix} \end{array} \right\}$$

As there are finitely many states and the cost is non-decreasing we know that at some point the set stabilizes s.t. $F_i = F_{i+1}$. We denote this set as $F_{final}$. Extracting a winning ML strategy, is now a check, asserting whether $(s_0, k) \in F_{final}$ for some $k$, and then returning a sequence of states starting at $s_0$ where the cost we need to pay to reach a final state is always decreasing.

## 4.2.1 Pruned Attractor Set

A side effect of the calculation of $F_i$, for any $i > 2$, is that we can have several pairs with the same state but a different cost. Consider the example in Figure 4.2 where Figure 4.2a is the game graph, and Table 4.2b shows the incremential computation of $F_{final}$ for the 1-WG $(G, AF_{\leq\infty}(Final))$.



**(a)** Game graph $G$.

| $F_i$ | Elements of $F_i$ |
|---|---|
| $F_0$ | $\{(s_3, 0)\}$ |
| $F_1$ | $\{(s_3, 0), (s_2, 0), (s_1, 2), (s_0, 4)\}$ |
| $F_2$ | $\{(s_3, 0), (s_2, 0), (s_1, 1), (s_1, 2), (s_0, 3), (s_0, 4)\}$ |
| $F_3$ | $\{(s_3, 0), (s_2, 0), (s_1, 1), (s_1, 2), (s_0, 2), (s_0, 3), (s_0, 4)\}$ |
| $F_4$ | $\{(s_3, 0), (s_2, 0), (s_1, 1), (s_1, 2), (s_0, 2), (s_0, 3), (s_0, 4)\}$ |

**(b)** Computation of $F_i$ for the game $(G, AF_{\leq\infty}(Final))$.

**Figure 4.2:** Example of a game where $Add_i$ causes pairs of the same state and different cost to be added to $F_i$

Having multiple pairs with the same state in $F_{final}$ is unnecessary, and the complexity of computing $F_{final}$ increases. Hence we introduce a method for continually computing a subset of $F_{final}$, that is, the subset only consisting of a state and the *least* cost. We start by defining the pruning function $P : \mathcal{P}(S \times \mathbb{N}_0) \rightarrow \mathcal{P}(S \times \mathbb{N}_0)$ where

$$P(F_i) = \{(s, k) \in F_i \mid \text{ there is no } j \text{ s.t. } j \leq k \text{ and } (s, j) \in F_i\}$$

For each calculation of $F_i$ a pruning is performed, such that only a single pair for each state is kept. We therefore define the pruned attractor set $F_i^P$ for all $i \in \mathbb{N}_0$ as

$$\begin{aligned} F_0^P &= \{(s,0) \mid s \in S_f\} \\ F_i^P &= P(F_{i-1} \cup Add_i) \end{aligned}$$

Again, consider the example in Figure 4.2, using the pruned attractor set method we get the following computation instead:

$$\begin{aligned} F_0^P =& \{(s_3,0)\} \\ F_1^P =& \{(s_3,0),(s_2,0),(s_1,2),(s_0,4)\} \\ F_2^P =& \{(s_3,0),(s_2,0),(s_1,1),(s_0,3)\} \\ F_3^P =& \{(s_3,0),(s_2,0),(s_1,1),(s_0,2)\} \\ F_4^P =& \{(s_3,0),(s_2,0),(s_1,1),(s_0,2)\} \end{aligned}$$

Notice, that the number of iterations does not decrease using the pruned attractor set method, however the space needed to store the attractor set decrease significantly.

### 4.2.2 Complexity of the attractor method

We observe that the addition of pairs with the smallest possible cost for a given state must appear regularly. We define such a pair as a minimal pair.

**Definition 4.5 (Minimal pair)**
A pair $(s,k)$ is minimal in some $F_i$ if for all $(s,k') \in F_i$ it holds that $k \leq k'$. We say $(s,k)$ is a *local minimal* pair if it is minimal in $F_i$ and $F_i \neq F_{final}$. Likewise, we say $(s,k)$ is a *global minimal* pair if it is minimal in $F_i$ and $F_i = F_{final}$.

Although a minimal pair is defined with respect to $F_i$, notice that it also holds for $F_i^P$. In fact, $F_i^p$ consists only of local minimal pairs of $F_i$.

**Lemma 8**
For any pair $(s,k) \in F_i \backslash F_{i-1}$, for some $i > 0$, there is a pair $(s',k') \in F_{i-1} \backslash F_{i-2}$ s.t. $s \xrightarrow{w} s'$ or $s \overset{w}{\dashrightarrow} s'$.

**Proof.** Follows from definition of $F_i$. Any pair in $F_i \backslash F_{i-1}$ was not added to $F_{i-1}$ because no such pair could satisfy the criteria of $Add_{i-1}$. □

We define the function $D_i : S \times \mathbb{N}_0 \to S \times \mathbb{N}_0$ for any $i > 0$ as:

$$D_i((s,k)) = \{(s',k') \mid (s',k') \in F_i \backslash F_{i-1} \text{ and } s \xrightarrow{w} s' \vee s \overset{w}{\dashrightarrow} s' \text{ and } w + k' \leq k\}$$

**Lemma 9**
Given $F_i = F_{i-1} \cup Add_i$ then whenever $F_i \backslash F_{i-1} \neq \emptyset$ then there is a pair $(s,k) \in F_i \backslash F_{i-1}$ s.t. $(s,k)$ is a global minimal pair.

**Proof.** We show that for any $F_i$, where $F_i \backslash F_{i-1} \neq \emptyset$, there is a global minimum pair. We do so by finding a local minimal pair in $F_i \backslash F_{i-1}$ and show that it is also a local minimum pair in $F_j$ for all $j > i$. First, we pick the pair $(s_{min}, k_{min}) \in F_i \backslash F_{i-1}$ s.t. $k_{min} \leq k'$ for all $(s', k') \in F_i \backslash F_{i-1}$. It follows from the definition of $Add_i$ that $(s_{min}, k_{min})$ is a local minimum pair in $F_i$. We now show that for any pair $(s,k) \in F_{i+1} \backslash F_i$, it holds that $k \geq k_{min}$. By Lemma 8 then given a pair $(s,k) \in F_{i+1} \backslash F_i$ there is a pair

$$(s_i, k_i) = \underset{(s',k') \in D_{i+1}((s,k))}{\arg\min} k'.$$

By definition of $Add_i$ it holds that $k_i \leq k$, however $k_{min} \leq k_i$, hence $k \not< k_{min}$. We therefore have that $(s_{min}, k_{min})$ is also a local minimum pair in $F_{i+1}$. By induction on $i$ we have that $(s_{min}, k_{min})$ is a local minimal pair in $F_j$, for all $j > i$. Eventually $F_j = F_{final}$ thus $(s_{min}, k_{min})$ is a global minimal pair. □

**Theorem 10 (Computing attractor set complexity)**
Given a 1-WG $(G, AF_{\leq c}\psi)$ and an initial state $s_0$, there is a polynomial time algorithm for computing a winning ML strategy $\sigma$, s.t. $G \upharpoonright \sigma, s_0 \models AF_{\leq c}\psi$.

**Proof.** Let $(G, AF_{\leq c}\psi)$ be a reachability game, where $G = (S, s_0, \mathcal{AP}, L, T_c, T_u)$ is a 1-WGG. We solve $(G, AF_{\leq c}\psi)$ by computing the attractor set $F_{final}$ and extracting the ML strategy. Computing any $F_i$ takes $O(|S| + |T|)$ where $T = T_c \cup T_u$. By Lemma 9, we have that in every iteration of $F_i$ we add at least one minimal pair. Therefore there can be at most $|S|$ iterations. Thus computing $F_{final}$ can be done in $O(|S| * (|S| + |T|)$.

Extracting the ML strategy is now simply a matter of checking whether $s_0$ is in $F_{final}$ and then choosing transitions s.t. the target state appears in a smaller $F_i$. This can be done in $O(|S| + |T|)$.

Hence computing a winning ML strategy takes $O(|S|^2 + |S| * |T| + |S| + |T|)$ time.

□

This method has the drawback, that the set $S_f$, must be known prior to the execution, disallowing an approach where the game graph is generated on-the-fly. We will therefore now consider a forward exploratory approach, with back-propagation of possible local answers.

## 4.3 Symbolic Dependency Graph

To find an on-the-fly algorithm for calculating a set of states with a winning strategy, we look at the DG framework. In 1998 Liu and Smolka presented a local on-the-fly minimum pre fixed-point algorithm for DGs[11]. A symbolic extension of DGs was presented by Jensen et al. [9] where they extend the local fixed-point algorithm for DGs into a minimum fixed-point algorithm for SDGs.

**Definition 4.6 (Symbolic dependency graph)**
A SDG is a tuple $D = (V, E, C)$ where:

- $V$ is a finite set of configurations,
- $E \subseteq V \times \mathcal{P}(\mathbb{N}_0 \times V)$ is a set of weighted hyper-edges, and
- $C \subseteq V \times \mathbb{N}_\infty \times V$ is a set of cover edges.

Let $D = (V, E, C)$ be an SDG. For a weighted hyper-edge, $e = (v, T) \in E$, we say $v$ is the source, $T$ is the target set, and $(w, u) \in T$ is a weighted hyper-edge branch with weight $w$ to the target $v$. From here on, we refer to weighted hyper-edges as hyper-edges. We define the function $succ : V \rightarrow E$ as the successor function where $succ(v) = \{(v, T) \in E\} \cup \{(v, w, u) \in C\}$ is the set of cover- and hyper-edges with $v$ as the source.

An assignment is a function $A : V \rightarrow \mathbb{N}_\infty$ assigning values to configurations of an SDG $D = (V, E, C)$. We denote the set of all assignments as $\mathcal{A}$ and the assignment $A(v) = \infty$ for all $v \in V$ as $A_\infty$. Furthermore, we define the partial order of assignments $\sqsubseteq$ as $A \sqsubseteq A'$ if and only if $A(v) \geq A'(v)$ for all $v \in V$. Note, $(\mathcal{A}, \sqsubseteq)$ is a complete-lattice.

The monotonic function $F : \mathcal{A} \rightarrow \mathcal{A}$ is defined as:

$$F(A)(v) = \begin{cases} 0 & \text{If exists } (v, w, u) \in C \text{ s.t. } A(u) \leq w < \infty \text{ or} \\ & A(u) < w = \infty \\ \min\limits_{(v,T) \in E} \left( \max\limits_{(w,u) \in T} (A(u) + w) \right) & \text{Otherwise} \end{cases}$$

We assume that $\max(\emptyset) = 0$ and $\min(\emptyset) = \infty$. A prefixed-point assignment $A \in \mathcal{A}$ is an assignment such that $F(A) \sqsubseteq A$. By Knaster-Tarskis fixed-point theorem we get that here exists a unique minimum prefixed-point assignment of an SDG, denoted $A_{min}$. Let $D = (V, E, C)$ be an SDG, then $A_{min}$ of $G$ can be computed by applying $F$ on $A_\infty$ finitely many times [9]. We refer to $F$ as the *global* algorithm.

Jensen et al. also presented an on-the-fly algorithm for computing a local minimum prefixed point assignment of an SDG[9]. We refer to this as the *local* algorithm and it has a pseudo polynomial time complexity. For further details we refer the reader to [9].

## 4.4 Dependency graph encoding of reachability games

In this section we show how to encode a 1-weighted reachability game into an SDG. Given a game $(G, AF_{\leq c}\psi)$ we construct the corresponding SDG rooted at $\langle s_0, AF_{\leq}\psi\rangle$. Figure 4.3 illustrate the hyper-edge and cover-edge rules for constructing the SDG. The upper-bound from the ReachWCTL$^u$ formula is encoded as the weight of the cover-edge going from $v_0$, see Figure 4.3a. Notice here that this method only supports a strict upper-bound, as $\infty$ has been assigned a different meaning. Below this point all configurations are of the form $\langle s\rangle$. The base case where the configuration $\langle s\rangle$ has a transition to the empty set is shown in Figure 4.3b.

The encoding of the remaining hyper-edges is then based on the same principle as computing the attractor set. There are two cases:

– When there are only uncontrollable transitions we encode all of them as a single hyper-edge as they will all need to be part of the winning strategy, illustrated in Figure 4.3c.

– In the case where there are controllable transitions we need to make sure that at least one of the controllable transitions and all of the uncontrollable transitions are part of a winning strategy. In order to achieve this each hyper-edge consists of one unique controllable transition and all of the uncontrollable transitions, see Figure 4.3d.

As a result of this encoding, propagating an assignment to configuration $\langle s\rangle$ intuitively corresponds to adding the state $s$ to the attractor set, giving us an obvious way to generate a winning strategy from the assignment.

**Theorem 11 (Encoding correctness)**
Let $(G, \varphi)$ be a 1-WG where $G = (S, s_0, \mathcal{AP}, L, T_c, T_u)$ and $\varphi = AF_{\leq c}\psi$. Let $D = (V, E, C)$ be the SDG constructed with root $\langle s_0, \varphi\rangle$. There exists a ML strategy $\sigma$ s.t. $G\restriction\sigma, s_0 \models_0 \varphi$, if and only if $A_{min}(\langle s_0, \varphi\rangle) = 0$.

The proof is future work, however we provide a small proof sketch.

**Proof sketch.** We prove Theorem 11 by structural induction on the formula. There are two cases:

*Formula is $AF_{\leq c}\psi$:* Show $A_{min}(\langle s, AF_{\leq c}\psi\rangle) = 0$ iff. $s \models AF_{\leq c}\psi$.

*Formula is $\psi$:* Let $\langle s\rangle$ be a configuration, then there are two cases:

*$s \models \psi$:* Trivial case. By Figure 4.3b and induction hypothesis we get that $s \models \psi$ iff $A_{min}(\langle s\rangle) = 0$.

*$s \not\models \psi$:* Again, there two cases, where $c' \leq c$:

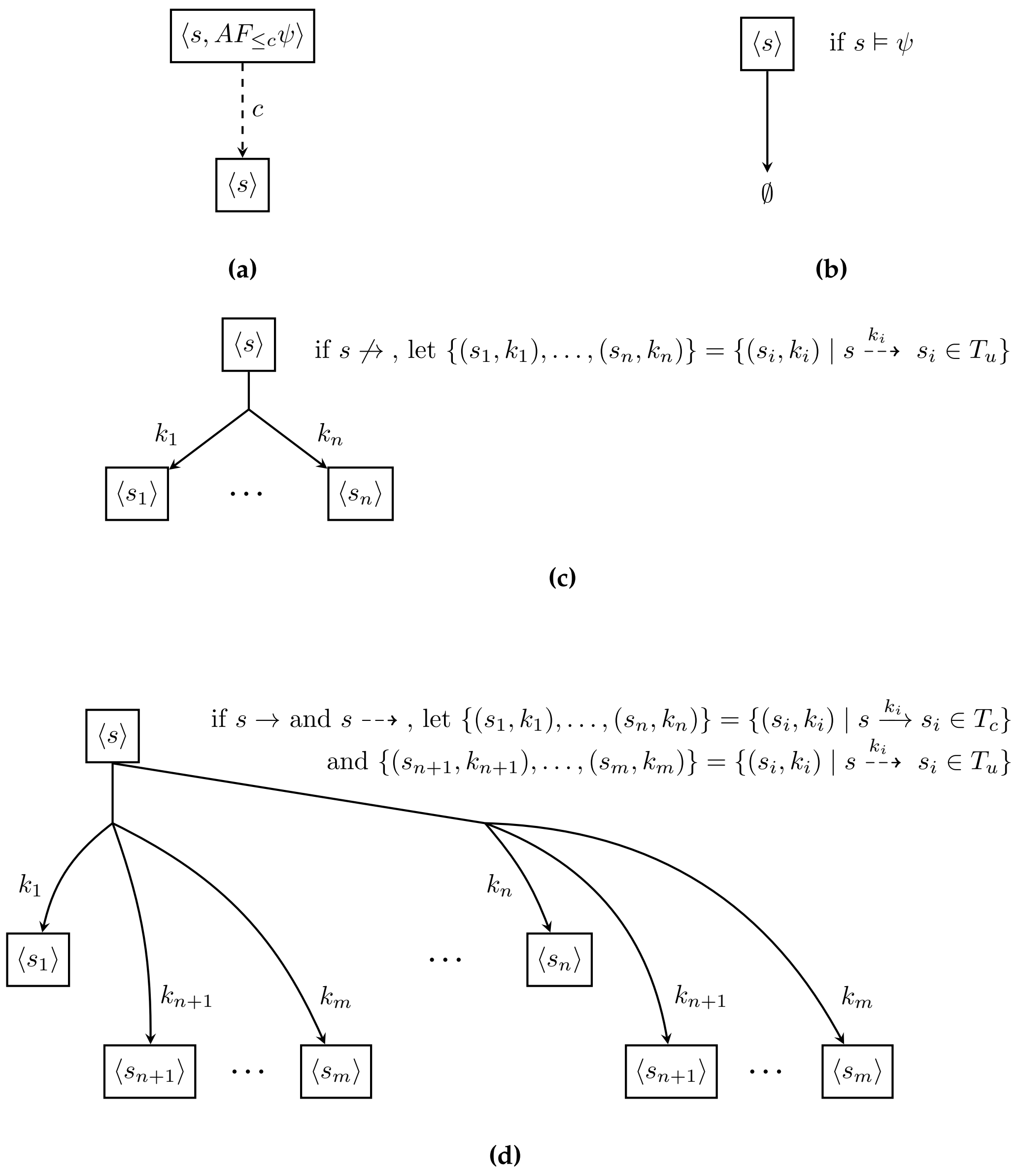

$\langle s, AF_{\leq c}\psi \rangle$
$c$
$\langle s \rangle$
(a)
$\langle s \rangle$
if $s \models \psi$
$\emptyset$
(b)
$\langle s \rangle$
if $s \not\rightarrow$ , let $\{(s_1, k_1), \ldots, (s_n, k_n)\} = \{(s_i, k_i) \mid s \overset{k_i}{\dashrightarrow} s_i \in T_u\}$
$k_1$
$k_n$
$\langle s_1 \rangle$ $\cdots$ $\langle s_n \rangle$
(c)
$\langle s \rangle$
if $s \rightarrow$ and $s \dashrightarrow$ , let $\{(s_1, k_1), \ldots, (s_n, k_n)\} = \{(s_i, k_i) \mid s \xrightarrow{k_i} s_i \in T_c\}$
and $\{(s_{n+1}, k_{n+1}), \ldots, (s_m, k_m)\} = \{(s_i, k_i) \mid s \overset{k_i}{\dashrightarrow} s_i \in T_u\}$
$k_1$
$k_n$
$k_{n+1}$
$k_m$
$k_{n+1}$
$k_m$
$\langle s_1 \rangle$ $\cdots$ $\langle s_n \rangle$
$\langle s_{n+1} \rangle$ $\cdots$ $\langle s_m \rangle$
$\langle s_{n+1} \rangle$ $\cdots$ $\langle s_m \rangle$
(d)

**Figure 4.3:** Dependency graph encoding of a 1-weighted reachability game $(G, AF_{\leq \bar{c}}\psi)$

*If $s \models AF_{\leq c'}\psi$ then $A_{min}(\langle s \rangle) = c'$:* Define $AF^{j}_{\leq c}\psi$ s.t. a state $s' \models AF^{0}_{\leq c}\psi$ iff $s' \models \psi$. Proof by induction on $j$.

*If $A_{min}(\langle s \rangle) = c'$ then $s \models AF_{\leq c'}\psi$:* Proof by induction.

Notice that the proof for case $s \not\models \psi$ includes both the edges from Figure 4.3c and 4.3d. □

## 4.5 Reachability Synthesis

As mentioned in the last section, there is a close correspondence between assigning a value to a configuration in an SDG and adding a state to an attractor set. In this section we present an algorithm, that based on this relationship, extracts a ML strategy from a minimum prefixed-point assignment of an SDG.

We start by defining a mapping between the constituents of an SDG and a 1-WG. Let $(G, AF_{\leq c}\psi)$ be a 1-WG, where $G = (S, s_0, \mathcal{AP}, L, T_c, T_u)$ is a 1-WGG and $D = (V, E, C)$ be an SDG constructed from $G$ by the rules presented in Section 4.4. We define a mapping between hyper-edges and controllable transitions as a function $ET : E \rightarrow T_c$ as:

$$ET(e = (v, T)) = \begin{cases} t = (v, k, u) & \text{if } t \in T_c \text{ and } (k, u) \in T \\ \textsc{nil} & \text{otherwise} \end{cases}$$

Note, by the rules of construction from Section 4.4, there can be only a single controllable transition which can satisfy the if condition of the $ET$ function.

The strategy extraction is presented in Algorithm 2. It takes four inputs: An SDG $D$, a root configuration $v_0 = \langle s_0, AF_{\leq c}\psi \rangle$, $A_{min}$ of $D$ and a 1-WG $G$ from which $D$ was constructed. Throughout its execution it maintains two data structures: A set of pairs $Q$ containing states and natural numbers, and a function $\Sigma$ which assigns controllable transitions to states. Upon termination the algorithm returns $\Sigma$. Notice that since the strategy is memoryless, it suffices to only map states to transitions. We write $\Sigma_{\textsc{nil}}$ when $\Sigma(s) = \textsc{nil}$ for any $s \in S$.

Algorithm 2 extracts the winning ML strategy based on the premise, that when a configuration in the SDG gets assigned a value less than infinity, then it is part of the attractor set. As such there exists a strategy from the corresponding state to a final state within some bound. The cover-edges act as guards, where its weight determines the maximal assigned value of its target configuration. If this value exceeds the weight of cover-edge, a ML strategy does not exist within the desired upper-bound. From here, Algorithm 2 deduces which hyper-edge propagated an assigned value, computes the controllable transition, if any such exists, and maps that to a state.

A winning ML strategy for a game can now be computed in the following manner: First, compute $A_{min}$ of an SDG constructed from the rules of Section 4.4.

**Algorithm 2** Strategy extraction algorithm for 1-weighted reachability games

**Input:** a SDG $D = (V, E, C)$, a root configuration $v_0 \in V$, an Assignment $A$ and a game $(G, AF_{\leq c}\psi)$.

**Output:** A winning strategy $\Sigma$ for $(G, AF_{\leq c}\psi)$ (if it exists).

```
1: $\Sigma \leftarrow \Sigma_{\text{NIL}}$
2: $Q \leftarrow \{v_0, 0\}$
3: while $Q \neq \emptyset$ do
4:     Pick $(v, k)$ from $Q$; $Q \leftarrow Q \backslash \{(v, k)\}$
5:     if $\exists e = (v, k', u) \in succ(v) \ \wedge \ A(v) = 0$ then
6:         $Q \leftarrow Q \cup \{(u, k)\}$
7:     else if $succ(v) \subseteq E$ then
8:         $e = (v, T) \leftarrow \operatorname*{arg\,min}_{(v,T') \in succ(v)} \max\{A(u) + k' \mid (k', u) \in T'\}$
9:         $\Sigma(v) \leftarrow ET(e)$
10:        $Q \leftarrow Q \cup \{(u, k + k') \mid (k', u) \in T\}$
11: return $\Sigma$
```

Second, extract the winning ML strategy using Algorithm 2.

**Remark 4**
The purpose of algorithm 2 is to show that synthesis can be done using minimum prefixed-point assignment computation of an SDG. Obviously computing the fixed-point assignment and extracting $\sigma$ can be done simultaneously, as such, in practice, synthesis should be done, by modifying the algorithm presented in [9].

## 4.6 Example: Synthesis Algorithm

We will now go through the process of synthesizing a winning ML strategy, using the SDG framework and Algorithm 2. Consider the example, from Section 3.4, with the self-driving car. We use a modified version of this example where we drop the first vector component from the game graph and the formula. The resulting formula is $\varphi = AF_{\leq 8}(\{Home\})$ and the resulting game graph is illustrated in Figure 4.4.

Consider the SDG constructed from this game by the rules in Figure 4.3 We can calculate the minimum pre-fixed point assignment using the local algorithm presented in [9] with the SDG as input. The resulting SDG is shown in Figure 4.5 where the assignment is illustrated as annotations to the configurations.

We will now go through the execution of Algorithm 2 with the annotated SDG as the input. Table 4.6 shows the content of $Q$ and the mapping $\Sigma$ for each iteration. Initially $Q = \{(v_0, 0)\}$ and $\Sigma$ maps to NIL for all states. From here we iterate over the content of $Q$ until it is empty. We begin by picking the pair $(v_0, 0)$. Then we

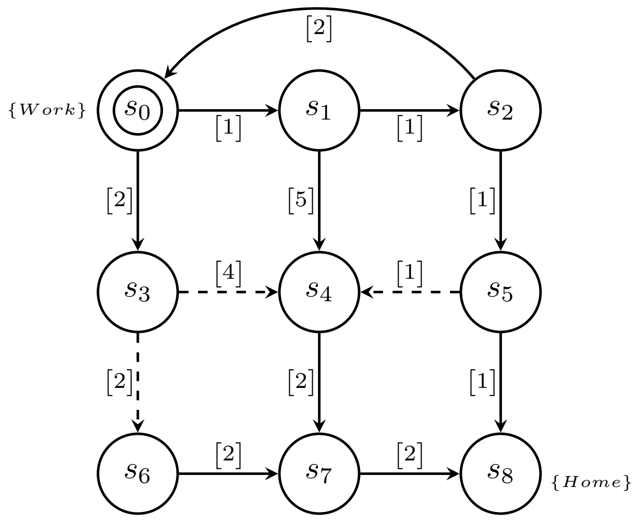


**Figure 4.4:** Modified game graph from Figure 3.2.

check whether or not $v_0$ is the source configuration of a cover-edge or a hyper-edge. As the initial configuration is always the source of a cover-edge, we enter the first if-clause of the algorithm. Here we check if the upper-bound holds by checking if the assigned value of the source is 0. We confirm this and add the pair $(\langle s_0 \rangle, 0)$ to $Q$ consisting of the target configuration and the unchanged cost.

In the second iteration we now pick the pair $(\langle s_0 \rangle, 0)$ and as $\langle s_0 \rangle$ has two outgoing hyper-edges we enter the second if-clause. We then pick the hyper-edge which propagated the assigned value of $\langle s_0 \rangle$ and map the corresponding controllable transition if it exists. In this case it is the hyper-edge $e = (\langle s_0 \rangle, \{(1, \langle s_1 \rangle)\})$, $t = (s_0, 1, s_1)$ and of course $\Sigma(s_0) = t$. The last thing we need to do is add pairs of all target configurations $e$ and their cost from $\langle s_0 \rangle$ to $Q$. We repeat this until the queue $Q$ is empty and output the ML strategy.

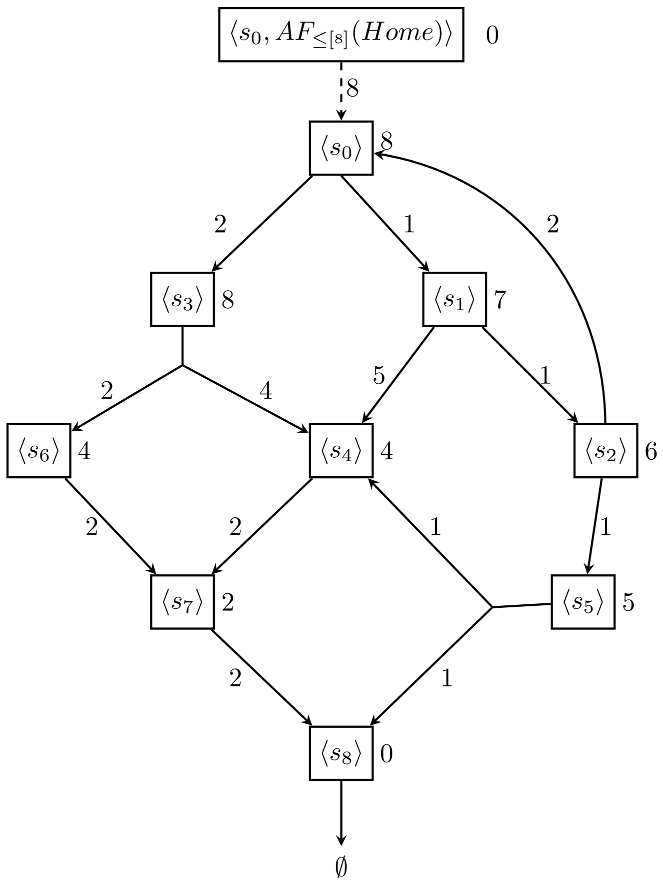


**Figure 4.5:** Annotated symbolic dependency graph of the game graph from Figure 4.4. The number to the right of each configuration is its $A_{min}$ value.

| Iteration | Q | Σ |
|---|---|---|
| 1 | $\{(\langle s_0, AF_{\leq 8}(Home)\rangle, 0)\}$ | $\Sigma = \Sigma_{\text{NIL}}$ |
| 2 | $\{(\langle s_0\rangle, 0)\}$ | $\Sigma(s_0) = (s_0, 1, s_1)$ |
| 3 | $\{(\langle s_1\rangle, 1)\}$ | $\Sigma(s_1) = (s_1, 1, s_2)$ |
| 4 | $\{(\langle s_2\rangle, 2)\}$ | $\Sigma(s_2) = (s_2, 1, s_5)$ |
| 5 | $\{(\langle s_5\rangle, 3)\}$ | $\Sigma(s_5) = (s_5, 1, s_8)$ |
| 6 | $\{(\langle s_8\rangle, 4), (\langle s_4\rangle, 4)\}$ | |
| 7 | $\{(\langle s_4\rangle, 4)\}$ | $\Sigma(s_4) = (s_4, 2, s_7)$ |
| 8 | $\{(\langle s_7\rangle, 6)\}$ | $\Sigma(s_7) = (s_7, 2, s_8)$ |
| 9 | $\{(\langle s_8\rangle, 8)\}$ | |

**Figure 4.6:** Algorithm 2 divided into rounds illustrating the construction of the ML strategy Σ

# Chapter 5

# Conclusion

We extended the *automata theoretic game* framework, by Thomas, by defining a two player game with multiple weights. With this extended framework we can express problems from domains with a multitude of discrete values. With the presented algorithms, we can then synthesize solutions for problems with a single discrete value. Besides extending the framework with weights, we also provide a hierarchy for the different strategy types. With this we found that a single-state cost strategy is sufficiently expressive in multiple weighted reachability games. We also found that a memoryless strategy is sufficiently expressive for 1-weighted reachability games. We then use this result to synthesize memoryless strategies for 1-weighted reachability games, where we prove polynomial time complexity for the problem and provide an on-the-fly algorithmic solution for the extraction. Additionally we present an undecidability result for model checking with the full WCTL logic and then present a decidable sub-logic, ReachWCTL$^u$, which express reachability properties in games.

## 5.1 Future work

Figure 5.1 shows an overview of possible areas of future works covering three major areas: Expanding logic, model formalism and prototype.

Summed up, we aim to have a fully functioning prototype for WCTL with upper- and lower-bounds. The prototype will include: A front-end (GUI) for writing games and presenting statistics, and a back-end for synthesising strategies.

We will now go through each branch of Figure 5.1 in more details in the following Subsections.

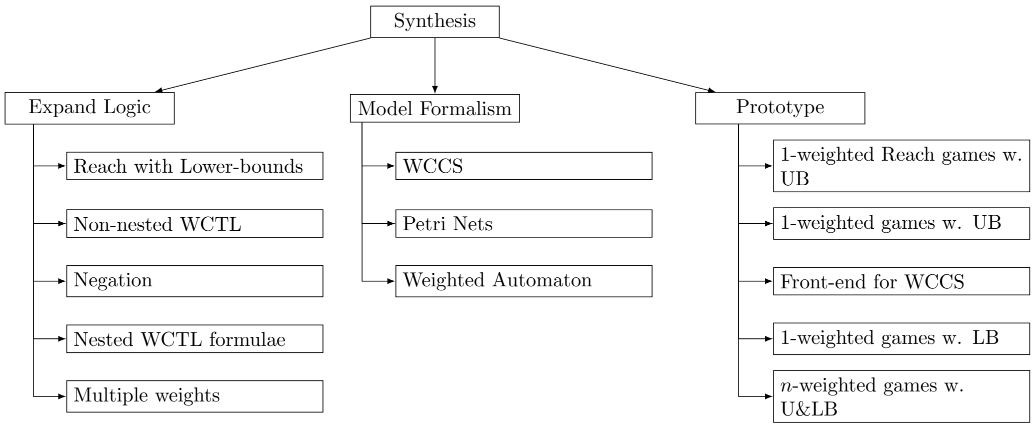


**Figure 5.1:** Future milestones of the project. A subset of these will be finished in the spring semester. Milestones are separated into three categories.

### 5.1.1 Expand Logic

We already proved that WCTL is undecidable (In Section 3.1), hence the future focus will be on strategies for lower- and upper-bounds. Proposition 2 showed, that a new type of strategy is necessary in order to synthesize applicable strategies from larger subsets of WCTL. As such, expanding the logic requires further research into the expressiveness of the strategies, and the synthesis methods: SDG framework and attractor set.

Another part of this area is multiple weights, shown in the very last branch of Figure 5.1. Currently, we have proven that a SSC strategy is sufficiently expressive to solve $n$-weighted reachability games. However, the synthesis methods (attractor set and SDG framework) needs to be extended to multiple weights. The obstacle here, is that the amount of incomparable cost vectors increase exponentially with the number of vector components. Furthermore as we expand the set of WCTL operators it is possible a new strategy type will have to be contrived.

### 5.1.2 Model Formalism

This branch is devoted to developing a model formalism which can express a large game in a compact and concise manner. Being able to express large $n$-WGs using a relative small, but expressive model, enhances the applicability. Initially we plan on extending the process algebra Weighted Calculus of Communicating Systems (WCCS) to a multi-weighted game variant. And possibly later, Petri Nets and Weighted Automatons.

### 5.1.3 Prototype

The area of prototyping involves, implementing and comparing the different synthesis methods presented in Sections 4.2 and 4.5. The current plan is to continue the development of WKTool [8]. WKTool currently includes all the necessary features to express 1-weighted WCCS processes along with various engines capable of verifying properties expressed in WCTL with upper-bounds (a subset of the WCTL presented in this paper). Preliminary work has already been put into a prototype for synthesis of strategies for 1-weighted games.

# Bibliography

[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). URL: `http://dblp.uni-trier.de/db/journals/lmcs/lmcs4.html#BouyerLM08`.

[2] Patricia Bouyer, Uli Fahrenberg, Kim G Larsen, Nicolas Markey, and Jiří Srba. "Infinite runs in weighted timed automata with energy constraints". In: *International Conference on Formal Modeling and Analysis of Timed Systems*. Springer. 2008, pp. 33–47.

[3] J. Richard Büchi and Lawrence H. Landweber. "Definability in the Monadic Second-Order Theory of Successor". In: *Journal of Symbolic Logic* 34.2 (1969), pp. 166–170.

[4] 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.

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

[6] E.Allen Emerson and Edmund M. Clarke. "Using branching time temporal logic to synthesize synchronization skeletons". In: *Science of Computer Programming* 2.3 (1982), pp. 241–266.

[7] Uli Fahrenberg, Line Juhl, Kim G Larsen, and Jiří Srba. "Energy games in multiweighted automata". In: *International Colloquium on Theoretical Aspects of Computing*. Springer. 2011, pp. 95–115.

[8] Jonas Finnemann Jensen and Lars Kaerlund Oestergaard. *WKTool*. 2014. URL: `http://github.com/jonasfj/WKTool`.

[9] 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.

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

[11] Xinxin Liu and Scott A. Smolka. "Simple Linear-Time Algorithms for Minimal Fixed Points (Extended Abstract)." In: *ICALP*. Ed. by Kim Guldstrand Larsen, Sven Skyum, and Glynn Winskel. Vol. 1443. Lecture Notes in Computer Science. Springer, 1998, pp. 53–66.

[12] A Pnueli, E Asarin, O Maler, and J Sifakis. "Controller synthesis for timed automata". In: *Proc. System Structure and Control. Elsevier*. Citeseer. 1998.

[13] Amir Pnueli and Roni Rosner. "On the synthesis of a reactive module". In: *Proceedings of the 16th ACM SIGPLAN-SIGACT symposium on Principles of programming languages*. ACM. 1989, pp. 179–190.

[14] Wolfgang Thomas. "On the Synthesis of Strategies in Infinite Games." In: *STACS*. Oct. 6, 2009, pp. 1–13. URL: `http://dblp.uni-trier.de/db/conf/stacs/stacs95.html#Thomas95`.