Differentiable Weighted Automata

Anand Balakrishnan [1] **Jyotirmoy V. Deshmukh** [1]

Abstract

Weighted automata have been used extensively for studying quantitative properties of systems, modeling the behavior of probabilistic systems, and various text, speech, and image processing domains. Moreover, they have become popular in reinforcement learning for task specifications in the form of reward machines. This is due to their ability to assign a quantitative score to some sequence of inputs in the problem domain. While the weighted automaton formalism is powerful and well-studied in literature, its inherently discrete structure makes it difficult to use in gradient-based pipelines. In this paper, we present a systematic framework for designing differentiable weighted automata that can leverage automatic differentiation tools to compute the gradient of the weight calculated by a weighted automaton with respect to its input sequence.

1. Introduction

Weighted automata are a generalization of finite-state automata over algebraic semiring structures that arose from the connection of finite automata and rational formal power series [1]–[3]. Here, in addition to checking if an input sequence (or word) is accepting or rejecting, a *weight* is associated with the input word and the *run* induced by input word in the weighted automaton. Weighted automata have extensively been used for the verification of quantitative properties [4]–[7], for reasoning about probabilistic systems [8], [9], and for text, speech, and image processing [10]–[13]. In recent literature on reinforcement learning, a specialization of weighted automata — *reward machines* — have been introduced to express reward functions for complex spatiotemporal tasks [14]–[18].

[1]Thomas Lord Department of Computer Science, University of Southern California, Los Angeles, CA 90005, USA. Correspondence to: Anand Balakrishnan <anandbal@usc.edu>, Jyotirmoy V. Deshmukh <jdeshmuk@usc.edu>.

Published at the 2^{nd} Differentiable Almost Everything Workshop at the 41^{st} International Conference on Machine Learning, Vienna, Austria. July 2024.

While weighted automata are powerful tools to compute quantitative costs or weights of structured input sequences — images, words, robot trajectories — they have been challenging to use in gradient-based pipelines due to their inherently discrete structures. For example, in trajectory optimization and control theory, automata-based techniques decompose the optimization problem over the automaton as "sub-tasks". Such frameworks usually reduce to a graph game on a composition of the specification automaton and an underlying model of the system [19], [20]; or as hierarchical hybrid systems [21]–[24]. In a similar vein, recent literature on reward machine-based reinforcement learning, various frameworks learn policies for such sub-tasks. The works in [25] and [26] propose compositional approaches to learning policies for sub-tasks in reward machines to satisfy a global task. These methods are inherently restricted use to the state explosion that happens by decomposing the automaton, and thus, don't scale well to more complex specifications and systems.

Our Contributions In this paper, we present a construction of weighted automata using matrix semirings that allows for the automaton to be directly deployed in neural network-based pipelines. We describe a set of necessary conditions to construct a class of weighted automata that can be *differentiable with respect to their inputs*, thereby leveraging state-of-the-art automatic differentiation tools [27]. Further, we present empirical results in trajectory optimization for some sequential, time-dependent tasks.

While a notion of differentiable weighted automata for natural language processing has been explored in [28], [29] — over graph operations on the automata, i.e., with respect to other automata or the transitions of an automaton — our approach looks at an orthogonal problem. Specifically, we study the semantics of differentiability of the weights outputted by a fixed automaton with respect to the input sequences — a common occurrence in trajectory optimization and reinforcement learning with reward machines. To the best of our knowledge, this is the first presentation of such differentiable automata along with the conditions to construct them.

2. Preliminaries

In this section, we will define some notations and provide some background for our proposed framework.

Definition 1 (Semiring [30])**.** The tuple $(K, \oplus, \otimes, \tilde{0}, \tilde{1})$ is a *semiring* with the underlying set K if $\left(K, \oplus, \tilde{0}\right)$ is a commutative monoid with identity $\tilde{0}$; $\left(K, \otimes, \tilde{1}\right)$ is a monoid with identity element $\tilde{1}$; $\otimes$ distributes over $\oplus$; and $\tilde{0}$ is an annihilator for $\otimes$ (for all $k \in K, k \otimes \tilde{0} = \tilde{0} \otimes k = \tilde{0}$).

Semirings are algebraic structures that are used to define the semantics of *non-deterministic weighted automata*, defined later in this section. Important examples of semirings include:

- Boolean semiring, $(\{0,1\}, \vee, \wedge, 0, 1)$ with *or*-operation acting as addition and *and*-operation acting as multiplication;
- The real numbers $(\mathbb{R}, +, \times, 0, 1)$ with the usual addition and multiplication.
- The log-semiring with the extended reals $(\mathbb{R} \cup \{-\infty, \infty\}, \oplus_{\log}, +, -\infty, 0)$, where $x \oplus_{\log} y = \log_b(b^x + b^y)$, for some base b;
- The tropical $\max$-plus semiring $(\mathbb{R} \cup \{-\infty, \infty\}, \max, +, -\infty, 0)$, which can be derived from the limit of log-semiring as the base $b \to \infty$.

The log-semiring and the tropical semiring are used extensively in the context of optimization, where the tropical semiring appears in shortest-path analysis (as the $\min$-plus semiring, which is isomorphic to the $\max$-plus semiring via negation). The former is used for smooth-approximations of $\min$ and $\max$ operations, and in problems involving log-likelihoods.

If the multiplication operation $\otimes$ is commutative, we say that the semiring is also commutative, and it is said to be *idempotent* if the addition operation is such that $x \oplus x = x$ for all $x \in K$. All of the above described semirings are commutative, while only the Boolean and tropical semirings are idempotent. In the following definitions, we will use K to denote semirings.

Definition 2 (Weighted Automaton [1])**.** A weighted automaton is a tuple $\mathcal{A} = (\Sigma, Q, Q_0, Q_F, \Delta\lambda, \alpha, beta)$, where

- Σ is an input alphabet;
- Q is a set of *locations* in the automaton, with Q_0 and Q_F denoting the initial and final (or accepting) locations respectively;
- $\Delta : Q \times \Sigma \to 2^Q$ is a transition function, where 2^Q denotes the powerset of Q;

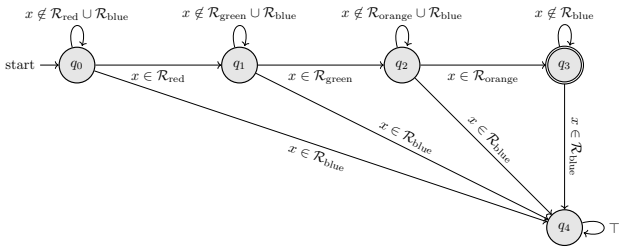

Figure 1. Example of an automaton with inputs $x \in \mathcal{X} \subset \mathbb{R}^n$. Here, for various regions $\mathcal{R}_{\text{red}}$, $\mathcal{R}_{\text{green}}$, $\mathcal{R}_{\text{orange}}$, $\mathcal{R}_{\text{blue}}$ contained in $\mathcal{X}$, the automaton specifies a task: "move to region $\mathcal{R}_{\text{red}}$, then region $\mathcal{R}_{\text{green}}$, and then to region $\mathcal{R}_{\text{orange}}$ in order while always avoiding the region $\mathcal{R}_{\text{blue}}$."

- $\lambda : Q \times \Sigma \times Q \to K$ is a transition weight function, $\alpha : Q_0 \to K$ is the initial weights function and $\beta : Q_F \to K$ is the final weights function.

As is convention, we use Σ^* to refer to the set of all finite length sequence of elements $x \in \Sigma$. Given an input sequence, $\xi = (x_0, x_1, \ldots, x_l) \in \Sigma^*$, a *run* in the symbolic automaton $\mathcal{A}$ is a sequence of locations $(q_0, q_1, \ldots, q_{l+1})$ such that $q_{i+1} \in \Delta(q_i, x_i)$ for all $i \in 0, \ldots, l$. We use $q_i \xrightarrow{x_i} q_{i+1}$ to denote a valid transition in the automaton, and $\mathsf{run}_{\mathcal{A}}(\xi)$ to denote the set of runs induced in $\mathcal{A}$ by $\xi \in \Sigma^*$. We say that the automaton is non-deterministic if a trace ξ can generate multiple runs in $\mathcal{A}$, i.e., if for some input x and location q, $|\Delta(q, x)| > 1$. A run is *accepting* in the automaton if $q_l \in Q_F$ for any induced run $(q_0, q_1, \ldots, q_l) \in \mathsf{run}_{\mathcal{A}}(\xi)$, and we denote this by $\xi \models \mathcal{A}$. The *weight* of an input trace $\xi = (x_0, \ldots, x_l) \in \Sigma^*$ on a weighted automaton $\mathcal{A}$ is

$$w_{\mathcal{A}}(\xi) = \bigoplus_{\mathsf{run}_{\mathcal{A}}(\xi)} \alpha(q_0) \otimes \left(\bigotimes_{i=0}^{l} \lambda(q_i, x_i, q_{i+1}) \right) \otimes \beta(q_{l+1}). \tag{1}$$

Example. Consider a 2D workspace $\mathcal{X} \subset \mathbb{R}^2$ with four regions $\mathcal{R}_{\text{red}}$, $\mathcal{R}_{\text{green}}$, $\mathcal{R}_{\text{orange}}$, $\mathcal{R}_{\text{blue}}$ contained in $\mathcal{X}$. Let us have a robot in the workspace that is required to visit the regions in the strict order $\mathcal{R}_{\text{red}}, \mathcal{R}_{\text{green}}, \mathcal{R}_{\text{orange}}$, while never entering the region $\mathcal{R}_{\text{blue}}$. In Figure 1, we see the (unweighted) automaton describing such a task: the input alphabet $\Sigma = \mathcal{X}$ and the expressions on the transitions (or *guards*) represent the condition when the transition is feasible, i.e., when the condition is true for a transition $q_i \xrightarrow{x} q_{i+1}$ then $q_{i+1} \in \Delta(q_i, x)$. Moreover, the location q_4 (with a tautology self-loop) is a *rejecting sink* location in the automaton. In Figure 2, we see an example of a trajectory that satisfies the requirement.

Definition 3 (Matrix Semirings [31])**.** If m is a positive integer and K is a semiring, then the set of $m \times m$ matrices with entries in K, denoted $K^{m \times m}$, is also a semiring.

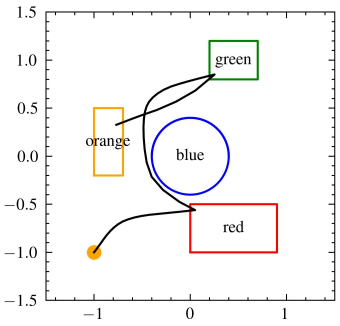

Figure 2. An example trajectory in a 2D workspace that satisfies the specification in Figure 1 described in Example 2. Here, the state x is a vector in $\mathcal{X} \subset \mathbb{R}^2$, and the trajectory starts at a point $(-1, -1)$ and completes the specified task.

Specifically, for matrices $A, B, C \in K^{m \times m}$, we can define the semiring operation as follows:

- Addition $A \oplus B = C$ is defined as element-wise addition $C_{ij} = A_{ij} \oplus B_{ij}$;
- The additive identity matrix is, intuitively, the $m \times m$ matrix with all entries $\tilde{0}$;
- Multiplication $AB = C$ is defined similar to matrix multiplication as $C_{ij} = \bigoplus_{k=0}^{m-1} A_{ik} \otimes B_{kj}$; and
- The multiplicative identity matrix (or simply, identity matrix) is similar to the usual: an $m \times m$ matrix with all diagonal entries equal to $\tilde{1}$, and the rest are $\tilde{0}$.

From the above definition of matrix semiring, one can derive the vector-dot product ($x_3 = x_1 \cdot x_2 = x_1 x_2$), and vector-matrix multiplication ($x_2 = x_1^T A$).[1] We refer readers to [31] to see various other linear algebra equivalences under the algebraic framework.

3. Differentiable Automata Operators

In this section, we will look at an alternative construction of weighted automata as *matrix operators* by leveraging extensive literature on automata theory has studied the field — especially weighted and quantitative automata — from the perspective of linear algebra [31]–[33].

Remark. Throughout the rest of the paper, we restrict ourselves to $\Sigma \subseteq \mathbb{R}^n$ in an effort to keep our framework relevant to compute pipelines that rely on floating point operations. In practice, any input alphabet can be encoded into some vector embedding, similar to what is done in image processing, natural language processing, and with graph neural networks. Moreover, we can use arbitrarily large numbers for semirings with ∞ to ease practical use.

[1] We will use the standard notation for matrix operations but defined over semirings unless otherwise specified.

Let the locations in Q an automaton $\mathcal{A}$ be indexed q_0 through $q_{|Q|-1}$. Then, we use $\mathbf{1}_{Q_0}$ and $\mathbf{1}_{Q_F}$ be the *Boolean indicator vectors* in $\{0,1\}^{|Q|}$ with entries such that

$$(\mathbf{1}_{Q_0})_i = \begin{cases} 0, & \text{if } q_i \notin Q_0 \\ 1, & \text{if } q_i \in Q_0 \end{cases}$$

$$(\mathbf{1}_{Q_F})_i = \begin{cases} 0, & \text{if } q_i \notin Q_F \\ 1, & \text{if } q_i \in Q_F. \end{cases}$$

Similarly, we define the initial weights $\boldsymbol{\alpha} \in K^{|Q|}$ and final weights $\boldsymbol{\beta} \in K^{|Q|}$ as

$$(\boldsymbol{\alpha})_i = \begin{cases} \tilde{0}, & \text{if } q_i \notin Q_0 \\ \alpha(q_i), & \text{if } q_i \in Q_0 \end{cases}$$

$$(\boldsymbol{\beta})_i = \begin{cases} \tilde{0}, & \text{if } q_i \notin Q_F \\ \beta(q_i), & \text{if } q_i \in Q_F. \end{cases}$$

Specifically, for a given weighted automaton $\mathcal{A}$ over a semiring K, we define alternative matrix operators $\mathbb{A}_\Delta : \Sigma \to \{0,1\}^{|Q| \times |Q|}$ and $\mathbb{A}_\lambda : \Sigma \to K^{|Q| \times |Q|}$, corresponding to Δ and λ respectively.

Definition 4 (Transition Matrix Operator)**.** For an automaton $\mathcal{A}$, the corresponding transition matrix operator $\mathbb{A}_\Delta : \Sigma \to \{0,1\}^{|Q| \times |Q|}$ is defined such that $\mathbb{A}_\Delta(x)$ is an element of the $|Q| \times |Q|$ Boolean matrix semiring, the entries of which are:

$$(\mathbb{A}_\Delta(x))_{ij} = \begin{cases} 0, & \text{if } q_j \notin \Delta(q_i, x) \\ 1, & \text{if } q_j \in \Delta(q_i, x), \end{cases} \tag{2}$$

where q_i and q_j are locations in Q indexed from 0 to $|Q| - 1$.

Definition 5 (Weight Matrix Operator)**.** For an automaton $\mathcal{A}$, the corresponding weight matrix operator $\mathbb{A}_\lambda : \Sigma \to K^{|Q| \times |Q|}$ is defined such that $\mathbb{A}_\lambda(x)$ is an element of the $|Q| \times |Q|$ K-matrix semiring, the entries of which are:

$$(\mathbb{A}_\lambda(x))_{ij} = \lambda(q_i, x, q_j), \tag{3}$$

where q_i and q_j are locations in Q indexed from 0 to $|Q| - 1$.

From the above and prior definitions, we can derive the following results:

Lemma 1. *Given an input $\xi = (x_0, x_1, \ldots, x_l) \in \Sigma^*$ and the function:*

$$\mathsf{Acc}_\mathcal{A}(\xi) = \mathbf{1}_{Q_0}^T \mathbb{A}_\Delta(x_0) \mathbb{A}_\Delta(x_1) \ldots \mathbb{A}_\Delta(x_l) \mathbf{1}_{Q_F}, \tag{4}$$

then $\xi \models \mathcal{A}$ if and only if $\mathsf{Acc}_\mathcal{A}(\xi) = 1$. Moreover, the weight $w_\mathcal{A}(\xi)$ is given by

$$w_\mathcal{A}(\xi) = \boldsymbol{\alpha}^T \mathbb{A}_\lambda(x_0) \mathbb{A}_\lambda(x_1) \ldots \mathbb{A}_\lambda(x_l) \boldsymbol{\beta} \tag{5}$$

Proposition 1 (Conditions for Differentiability)**.** *Given an automaton $\mathcal{A}$ and a semiring $K \subseteq \mathbb{R}$, the gradient of the weight with respect to input sequences $\xi \in \Sigma^*$, $\frac{\partial w_{\mathcal{A}}(\xi)}{\partial \xi}$, is well-defined if and only if:*

- *$a_1 \oplus a_2$ and $a_1 \otimes a_2$ are differentiable at all points in Σ, for $a_1, a_2 \in K$;*
- *$\lambda(q, x, q')$ is differentiable at all $x \in \Sigma$ for all feasible $q \xrightarrow{x} q'$ transitions in $\mathcal{A}$.*

Example. In the example in Figure 1, by using the log-semiring with the weighting function $\lambda(q, x, q')$ as the Euclidean distance of x from the regions that enable each transition in the automaton in Figure 1, we can obtain a differentiable $w_{\mathcal{A}}$. Moreover, since both, $\oplus_{\log}$ and Euclidean distance are smooth functions, we can see that optimizing $w_{\mathcal{A}}$ — a smooth function — gives an approximation of the shortest path in the system that satisfies the specification.

4. Experimental Results

To demonstrate the use of differentiable weighted automata as specifications for trajectory optimization, we look at the example of two different environments. For both these environments, we use JAX [34] and Brax [35] to simulate the dynamics of the agent in the environment, and the log-semiring to compute the weights of the trajectories. Moreover, since we look at the log-semiring, our goal is to *maximize* the weights of the trajectory, as results in [36] establish that a non-negative weight for the log-semiring is equivalent to finding a satisfying input sequence. The results of our experiments are detailed in Table 1, where we see the number of gradient update epochs required to find a satisfying trajectory in the environment with the corresponding learning rate.[2] We set the control inputs over the entire discrete-time trajectory in both environments as the optimization parameters.

Unicycle We perform control of a second-order unicycle model on a 2-dimensional workspace, where the state vector of the agent consists of the position, heading, velocity, and angular velocity; we control the linear and angular acceleration of the unicycle; and the task at hand is the sequential reach-avoid automaton in Figure 1. For each transition guard on the automaton, the distance function λ computes the distance from the current position of the unicycle to the condition regions. The system is sampled at 0.1 second rate for a trajectory of length 80 time steps (or 8 seconds).

Pusher The *Pusher* environment in Brax [35] is a 23-dimensional system where we control a multi-jointed arm to push an object from its initial location to a goal location within 1000 time steps (at a period of 0.05 seconds). The automaton in Figure 3 describes this specification, where the predicates holdObj and reachGoal correspond to the task for the arm reaching and holding the object, and then subsequently for the object to reach the goal. To make the task of "reaching" a location feasible, we describe them as follows:

$$\begin{aligned} \mathsf{holdObj} &\iff \|p_{arm} - p_{obj}\| \leq 0.005\text{m} \\ \mathsf{reachGoal} &\iff \|p_{obj} - p_{goal}\| \leq 0.005\text{m}, \end{aligned}$$

where p_{arm}, p_{obj}, and p_{goal} are the 3D coordinates of the end of the arm, the object, and the goal position, respectively. Moreover, for each transition guard in the automaton, we pick the weight function λ to be as follows:

$$\begin{aligned} \mathsf{holdObj} &: \lambda(\cdot, x, \cdot) = -\max(0, \|p_{arm} - p_{obj}\| - 0.005) \\ \neg\mathsf{holdObj} &: \lambda(\cdot, x, \cdot) = -\max(0, 0.005 - \|p_{arm} - p_{obj}\|) \\ \mathsf{reachGoal} &: \lambda(\cdot, x, \cdot) = -\max(0, \|p_{obj} - p_{goal}\| - 0.005) \\ \top &: \lambda(\cdot, x, \cdot) = \tilde{0} = -\infty \end{aligned}$$

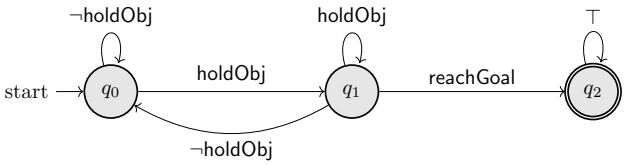

Figure 3. The automaton for the *Pusher* environment, where an articulated arm is used to push an object from one position to a goal location. Here, the predicate holdObj specifies that the arm must reach/hold the object, and reachGoal specifies that the object must reach the goal location.

Table 1. Results for Gradient-based Trajectory Optimization

Environment	Epochs to satisfaction	Learning Rate
Unicycle	425	0.05
Pusher	1562	0.05

5. Conclusion

In this paper, we present a framework for designing differentiable weighted automata. To the best of our knowledge, this is the first such paper to propose a set of necessary conditions for differentiability of a weighted automaton output with respect to the input sequence. We present an initial set of experiments to demonstrate the use of our framework in a gradient-based trajectory optimization problem, both examples being non-linear systems with sequential tasks.

[2] As these environments don't have highly non-convex objectives, simple gradient ascent is sufficient, but, in general, one must choose an appropriate optimizer and set of hyperparameters.

References

[1] M. Droste, W. Kuich, and H. Vogler, Eds., *Handbook of Weighted Automata* (Monographs in Theoretical Computer Science. An EATCS Series). Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, ISBN: 978-3-642-01492-5. DOI: 10.1007/978-3-642-01492-5.

[2] U. Boker, "Quantitative vs. Weighted Automata," in *Reachability Problems*, P. C. Bell, P. Totzke, and I. Potapov, Eds., vol. 13035, Cham: Springer International Publishing, 2021, pp. 3–18, ISBN: 978-3-030-89716-1. DOI: 10.1007/978-3-030-89716-1_1.

[3] A. Salomaa and M. Soittola, *Automata-Theoretic Aspects of Formal Power Series*. New York, NY: Springer New York, 1978, ISBN: 978-1-4612-6264-0. DOI: 10.1007/978-1-4612-6264-0.

[4] K. Chatterjee, L. Doyen, and T. A. Henzinger, "Alternating Weighted Automata," in *Fundamentals of Computation Theory*, M. Kutyłowski, W. Charatonik, and M. Gębala, Eds., Berlin, Heidelberg: Springer, 2009, pp. 3–13, ISBN: 978-3-642-03409-1. DOI: 10.1007/978-3-642-03409-1_2.

[5] K. Chatterjee, T. A. Henzinger, and J. Otop, "Quantitative Automata under Probabilistic Semantics," in *Proceedings of the 31st Annual ACM/IEEE Symposium on Logic in Computer Science*, ser. LICS '16, New York, NY, USA: Association for Computing Machinery, Jul. 2016, pp. 76–85, ISBN: 978-1-4503-4391-6. DOI: 10.1145/2933575.2933588.

[6] M. Droste and P. Gastin, "Weighted Automata and Weighted Logics," *Theoretical Computer Science*, Automata, Languages and Programming, vol. 380, no. 1, pp. 69–86, Jun. 2007, ISSN: 0304-3975. DOI: 10.1016/j.tcs.2007.02.055.

[7] R. Alur, L. D'Antoni, J. V. Deshmukh, M. Raghothaman, and Y. Yuan, *Regular Functions, Cost Register Automata, and Generalized Min-Cost Problems*, Feb. 2012. DOI: 10.48550/arXiv.1111.0670. arXiv: 1111.0670 [cs].

[8] K. Chatterjee, L. Doyen, and T. A. Henzinger, "Probabilistic Weighted Automata," in *CONCUR 2009 - Concurrency Theory*, M. Bravetti and G. Zavattaro, Eds., Berlin, Heidelberg: Springer, 2009, pp. 244–258, ISBN: 978-3-642-04081-8. DOI: 10.1007/978-3-642-04081-8_17.

[9] S. Adhikary, S. Srinivasan, J. Miller, G. Rabusseau, and B. Boots, "Quantum Tensor Networks, Stochastic Processes, and Weighted Automata," in *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics*, PMLR, Mar. 2021, pp. 2080–2088.

[10] K. Knight and J. Graehl, "An Overview of Probabilistic Tree Transducers for Natural Language Processing," in *Computational Linguistics and Intelligent Text Processing*, D. Hutchison, T. Kanade, J. Kittler, *et al.*, Eds., vol. 3406, Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 1–24, ISBN: 978-3-540-30586-6. DOI: 10.1007/978-3-540-30586-6_1.

[11] K. Knight and J. May, "Applications of Weighted Automata in Natural Language Processing," in *Handbook of Weighted Automata*, M. Droste, W. Kuich, and H. Vogler, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 571–596, ISBN: 978-3-642-01492-5. DOI: 10.1007/978-3-642-01492-5_14.

[12] M. Mohri, F. Pereira, and M. Riley, *Weighted Automata in Text and Speech Processing*, Mar. 2005. DOI: 10.48550/arXiv.cs/0503077. arXiv: cs/0503077.

[13] K. Culik and J. Kari, "Image Compression Using Weighted Finite Automata," *Computers & Graphics*, vol. 17, no. 3, pp. 305–313, May 1993, ISSN: 0097-8493. DOI: 10.1016/0097-8493(93)90079-O.

[14] R. T. Icarte, T. Klassen, R. Valenzano, and S. McIlraith, "Using Reward Machines for High-Level Task Specification and Decomposition in Reinforcement Learning," in *Proceedings of the 35th International Conference on Machine Learning*, PMLR, Jul. 2018, pp. 2107–2116. [Online]. Available: https://proceedings.mlr.press/v80/icarte18a.html.

[15] R. Toro Icarte, E. Waldie, T. Klassen, R. Valenzano, M. Castro, and S. McIlraith, "Learning Reward Machines for Partially Observable Reinforcement Learning," in *Advances in Neural Information Processing Systems*, vol. 32, Curran Associates, Inc., 2019. [Online]. Available: https://proceedings.neurips.cc/paper/2019/hash/532435c44bec236b471a47a88d63513d-Abstract.html.

[16] A. Camacho, R. Toro Icarte, T. Q. Klassen, R. Valenzano, and S. A. McIlraith, "LTL and Beyond: Formal Languages for Reward Function Specification in Reinforcement Learning," in *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence*, Macao, China: International Joint Conferences on Artificial Intelligence Organization, Aug. 2019, pp. 6065–6073, ISBN: 978-0-9992411-4-1. DOI: 10.24963/ijcai.2019/840.

[17] J. Corazza, I. Gavran, and D. Neider, "Reinforcement Learning with Stochastic Reward Machines," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 36, no. 6, pp. 6429–6436, Jun. 2022, ISSN: 2374-3468. DOI: 10.1609/aaai.v36i6.20594.

[18] A. Balakrishnan, S. Jakšić, E. A. Aguilar, D. Ničković, and J. V. Deshmukh, "Model-Free Reinforcement Learning for Spatiotemporal Tasks Using Symbolic Automata," in *2023 62nd IEEE Conference on Decision and Control (CDC)*, Dec. 2023, pp. 6834–6840. DOI: 10.1109/CDC49753.2023.10383559.

[19] C. Baier and J.-P. Katoen, *Principles of Model Checking*. MIT Press, Apr. 2008, ISBN: 978-0-262-30403-0.

[20] T. Wongpiromsarn, U. Topcu, and R. M. Murray, "Receding Horizon Temporal Logic Planning," *IEEE Transactions on Automatic Control*, vol. 57, no. 11, pp. 2817–2830, Nov. 2012, ISSN: 1558-2523. DOI: 10.1109/TAC.2012.2195811.

[21] H. Kress-Gazit, G. E. Fainekos, and G. J. Pappas, "Temporal-Logic-Based Reactive Mission and Motion Planning," *IEEE Transactions on Robotics*, vol. 25, no. 6, pp. 1370–1381, Dec. 2009, ISSN: 1941-0468. DOI: 10.1109/TRO.2009.2030225.

[22] G. E. Fainekos, A. Girard, H. Kress-Gazit, and G. J. Pappas, "Temporal Logic Motion Planning for Dynamic Robots," *Automatica*, vol. 45, no. 2, pp. 343–352, Feb. 2009, ISSN: 0005-1098. DOI: 10.1016/j.automatica.2008.08.008.

[23] S. L. Smith, J. Tůmová, C. Belta, and D. Rus, "Optimal Path Planning for Surveillance with Temporal-Logic Constraints," *The International Journal of Robotics Research*, vol. 30, no. 14, pp. 1695–1708, Dec. 2011, ISSN: 0278-3649. DOI: 10.1177/0278364911417911.

[24] L. Lindemann and D. V. Dimarogonas, "Efficient Automata-based Planning and Control under Spatio-Temporal Logic Specifications," in *2020 American Control Conference (ACC)*, Jul. 2020, pp. 4707–4714. DOI: 10.23919/ACC45564.2020.9147796.

[25] K. Jothimurugan, R. Alur, and O. Bastani, "A composable specification language for reinforcement learning tasks," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[26] L. Illanes, X. Yan, R. T. Icarte, and S. A. McIlraith, "Symbolic Plans as High-Level Instructions for Reinforcement Learning," *Proceedings of the International Conference on Automated Planning and Scheduling*, vol. 30, pp. 540–550, Jun. 2020, ISSN: 2334-0843. [Online]. Available: https://ojs.aaai.org/index.php/ICAPS/article/view/6750.

[27] A. G. Baydin, B. A. Pearlmutter, A. A. Radul, and J. M. Siskind, "Automatic Differentiation in Machine Learning: A Survey," *Journal of Machine Learning Research*, vol. 18, no. 153, pp. 1–43, 2018, ISSN: 1533-7928. [Online]. Available: http://jmlr.org/papers/v18/17-468.html.

[28] A. Hannun, V. Pratap, J. Kahn, and W.-N. Hsu. "Differentiable Weighted Finite-State Transducers." arXiv: 2010.01003 [cs, stat]. (Oct. 2, 2020), preprint.

[29] A. Hannun, "An Introduction to Weighted Automata in Machine Learning," 2021. [Online]. Available: https://awnihannun.com/writing/automata_ml/automata_in_machine_learning.pdf.

[30] M. Mohri, "Semiring Frameworks and Algorithms for Shortest-Distance Problems," *Journal of Automata, Languages and Combinatorics*, vol. 7, no. 3, pp. 321–350, Jan. 2002, ISSN: 1430-189X.

[31] J. S. Golan, *Semirings and Affine Equations over Them: Theory and Applications*. Dordrecht: Springer Netherlands, 2003, ISBN: 978-94-017-0383-3. DOI: 10.1007/978-94-017-0383-3.

[32] W. Kuich and A. Salomaa, *Semirings, Automata, Languages*. Berlin, Heidelberg: Springer Berlin Heidelberg, 1986, ISBN: 978-3-642-69959-7.

[33] M. Mohri, "Weighted Automata Algorithms," in *Handbook of Weighted Automata*, M. Droste, W. Kuich, and H. Vogler, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 213–254, ISBN: 978-3-642-01492-5. DOI: 10.1007/978-3-642-01492-5_6.

[34] R. Frostig, M. J. Johnson, and C. Leary, "Compiling machine learning programs via high-level tracing," *Systems for Machine Learning*, vol. 4, no. 9, 2018. [Online]. Available: https://mlsys.org/Conferences/doc/2018/146.pdf.

[35] C. D. Freeman, E. Frey, A. Raichuk, S. Girgin, I. Mordatch, and O. Bachem, *Brax – A Differentiable Physics Engine for Large Scale Rigid Body Simulation*, Jun. 2021. DOI: 10.48550/arXiv.2106.13281. arXiv: 2106.13281 [cs].

[36] S. Jakšić, E. Bartocci, R. Grosu, and D. Ničković, "An Algebraic Framework for Runtime Verification," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 37, no. 11, pp. 2233–2243, Nov. 2018, ISSN: 1937-4151. DOI: 10.1109/TCAD.2018.2858460.