Neural networks as fuzzy logic formulas

Damian Heiman[1], Antti Kuusisto[1] and Esko Turunen[1]

[1]Mathematics Research Centre, Tampere University, Korkeakoulunkatu 7, Tampere, 33720, Finland

Abstract

Neural networks are a fundamental aspect of modern artificial intelligence, playing a key role in various important machine learning architectures including transformers and graph neural networks. Recently, logical characterisations have been used to study the expressive power of many machine learning architectures, but there have been surprisingly few characterisations of plain neural networks. In this paper, we provide fuzzy logic characterisations of rational-weight ReLU-activated neural networks where the activation value of a neuron can be any real number via four fuzzy logics: Rational Pavelka logic (RPL), an extension of RPL called $\mathrm{RPL}(\odot)_{\leq 1}$ and two fragments of $L\Pi\frac{1}{2}$ called $L\Pi\frac{1}{2}(\to_P^-)_{\leq 1}$ and $L\Pi\frac{1}{2}(\odot^-, \to_P^-)$. We also provide fuzzy logic characterisations of a generalised polynomial ring over $\mathbb{Q}$ in countably many variables where the use of the ReLU-function is permitted via two fuzzy logics: $\mathrm{RPL}(\odot)$ and a fragment of $L\Pi\frac{1}{2}$ called $L\Pi\frac{1}{2}(\to_P^-)$.

Keywords

Neural networks, fuzzy logic, logical characterisations, polynomial rings

1. Introduction

Neural networks are directed graphs where each vertex (or neuron) obtains an activation value (a real number) either as an input or by aggregating the activation values of its in-neighbours. The aggregation scheme we consider is a standard one, i.e., we take a weighted sum of the activation values of in-neighbours, add a bias and apply the activation function $\mathrm{ReLU}(x) = \max\{0, x\}$. The neural networks we consider are feedforward (i.e., there are no directed cycles in the graph) and fully connected (i.e., the neurons are partitioned into layers, and from each neuron of a layer there is an edge to exactly every neuron of the next layer), and the weights and biases are rational numbers. An illustration of a neural network is given in Figure 1.

On the formal level, we identify a neuron with its aggregation formula broken down recursively to the neurons on the first layer and we subsequently identify a neural network with the tuple of neurons on the last layer. This makes neurons a special case of polynomials of the polynomial ring $\mathbb{Q}[x_i]_{i\in\mathbb{N}}$ (i.e., the ring of polynomials over rational numbers in countably many variables) extended with the ReLU-function. We refer to these ReLU-polynomials as terms, and we call the collection of these terms $\mathbb{Q}[\mathrm{ReLU}, x_i]_{i\in\mathbb{N}}$. Neural networks are a subclass of tuples of terms of $\mathbb{Q}[\mathrm{ReLU}, x_i]_{i\in\mathbb{N}}$.

We study the expressive power of $\mathbb{Q}[\mathrm{ReLU}, x_i]_{i\in\mathbb{N}}$ and of neural networks via two fuzzy logics that can express rational numbers and multiplication. Fuzzy logics generalise ordinary Boolean logics by allowing the truth value of a formula to be any value in the closed interval $[0, 1]$. For example, the infinite-valued Łukasiewicz logic [1] is simply a fuzzy version of Boolean propositional logic. We consider two families of fuzzy logics that extend Łukasiewicz logic. First, Rational Pavelka Logic (RPL) [2, 3, 4, 5] extends Łukasiewicz logic by adding a truth constant for every rational number between 0 and 1, and $\mathrm{RPL}(\odot)$ [6] extends RPL with the product conjunction $\odot$. On the other hand, $L\Pi\frac{1}{2}$ [7]

LOGICNN 2026: Workshop on Logical Methods for Neural Network Analysis, July 25, 2026, Lisbon, Portugal
damian.heiman@tuni.fi (D. Heiman); antti.kuusisto@tuni.fi (A. Kuusisto); esko.turunen@tuni.fi (E. Turunen)

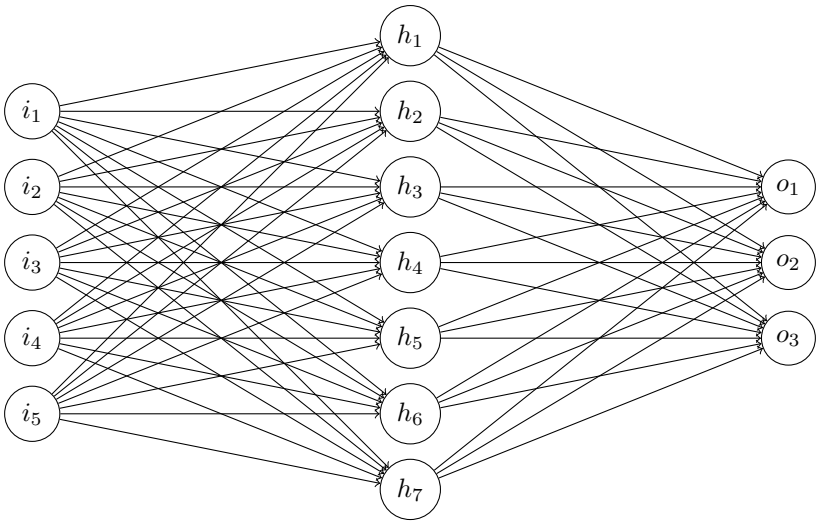

Figure 1: A visualization of a neural network consisting of three layers: an *input layer* $(i_1, \ldots, i_5)$, a *hidden layer* $(h_1, \ldots, h_7)$ and an *output layer* (o_1, o_2, o_3).

extends Łukasiewicz logic by combining it with Product Logic (adding the product conjunction $\odot$ and the product implication $\to_P$) and adding the truth constant $\overline{\frac{1}{2}}$. The logic $L\Pi\frac{1}{2}$ allows the construction of formulae equivalent to rational numbers between 0 and 1 by using the product implication $\to_P$ which corresponds to division. However, the connective $\to_P$ also increases the expressive power of $L\Pi\frac{1}{2}$ beyond that of $\mathrm{RPL}(\odot)$. We therefore restrict to the fragment $L\Pi\frac{1}{2}(\to_P^-)$ which limits $L\Pi\frac{1}{2}$ by not allowing proposition symbols in the scope of $\to_P$.

As the value of a term of $\mathbb{Q}[\mathrm{ReLU}, x_i]_{i\in\mathbb{N}}$ and the activation value of a neuron is allowed to be any real number and the truth values of fuzzy logic lie between 0 and 1, it is obviously impossible in the general case to find a formula of fuzzy logic whose truth values would be identical to the values of a given term of $\mathbb{Q}[\mathrm{ReLU}, x_i]_{i\in\mathbb{N}}$. Therefore, we naturally scale the activation values down to the interval $[0, 1]$ when translating to the logics. This is the most technical part of the translation, as multiplication behaves differently in the interval $[0, 1]$ compared to arbitrary intervals $[-k, k]$. No scaling is necessary in the opposite direction. In our characterisations, we translate tuples of terms to tuples of formulae and vice versa, rather than simply translating terms to formulae and vice versa. Our first theorem is as follows:

Theorem 4.1. *$\mathbb{Q}[\mathrm{ReLU}, x_i]_{i\in\mathbb{N}}$ has the same expressive power (w.r.t. scaling) as $\mathrm{RPL}(\odot)$ and $L\Pi\frac{1}{2}(\to_P^-)$.*

Theorem 4.1 also holds when we restrict the degrees of the ReLU-polynomials and apply a corresponding restriction on the logic side. This involves defining fragments $\mathrm{RPL}(\odot)_{\leq d}$ and $L\Pi\frac{1}{2}(\to_P^-)_{\leq d}$ of $\mathrm{RPL}(\odot)$ and $L\Pi\frac{1}{2}(\to_P^-)$, respectively, which limit how proposition symbols are allowed to appear in the scope of the product conjunction $\odot$ (the details can be found in Section 2.2.5). We obtain the following result:

Theorem 4.2. *For each $d \in \mathbb{N}$, $\mathbb{Q}[\mathrm{ReLU}, x_i]_{i\in\mathbb{N}}$ with degree bound d has the same expressive power (w.r.t. scaling) as $\mathrm{RPL}(\odot)_{\leq d}$ and $L\Pi\frac{1}{2}(\to_P^-)_{\leq d}$.*

When characterising neural networks, we consider on the one hand the fragments $\mathrm{RPL}(\odot)_{\leq 1}$ and $L\Pi\frac{1}{2}(\to_P^-)_{\leq 1}$, which restrict the respective logics $\mathrm{RPL}(\odot)$ and $L\Pi\frac{1}{2}(\to_P^-)$ by demanding that all proposition symbols in the scope of a product conjunction $\odot$ appear on the same side of the connective. On the other hand, we also consider RPL and $L\Pi\frac{1}{2}(\odot^-, \to_P^-)$, where $L\Pi\frac{1}{2}(\odot^-, \to_P^-)$ limits both $\odot$ and $\to_P$ by not allowing proposition symbols in the scope of either connective. We find that we can translate $\mathrm{RPL}(\odot)_{\leq 1}$ and $L\Pi\frac{1}{2}(\to_P^-)_{\leq 1}$ into neural networks and that we can translate neural networks into RPL and $L\Pi\frac{1}{2}(\odot^-, \to_P^-)$. Our third result is then given as follows:

Table 1
A summary of our results. The symbol $\equiv$ denotes same expressive power (subject to scaling when translating from left to right).

$\mathbb{Q}[\mathrm{ReLU}, x_i]_{i\in\mathbb{N}} \equiv \mathrm{RPL}(\odot)$, $L\Pi\frac{1}{2}(\to_P^-)$
$\mathbb{Q}[\mathrm{ReLU}, x_i]_{i\in\mathbb{N}}$ with degree bound $d \equiv \mathrm{RPL}(\odot)_{\leq d}$, $L\Pi\frac{1}{2}(\to_P^-)_{\leq d}$
Neural networks $\equiv \mathrm{RPL}(\odot)_{\leq 1}$, $L\Pi\frac{1}{2}(\to_P^-)_{\leq 1}$, RPL, $L\Pi\frac{1}{2}(\odot^-, \to_P^-)$

Theorem 4.5. *Neural networks have the same expressive power (w.r.t. scaling) as the logics* $\mathrm{RPL}(\odot)_{\leq 1}$, $L\Pi\frac{1}{2}(\to_P^-)_{\leq 1}$, RPL *and* $L\Pi\frac{1}{2}(\odot^-, \to_P^-)$.

Note that in the results listed above, the notion of expressive power is tied to scaling; it does *not* mean that we could, e.g., translate each formula φ of $\mathrm{RPL}(\odot)_{\leq 1}$ into a formula φ' of RPL such that φ' would always receive the same truth value as φ. However, it *does* follow that we could obtain a formula φ' of RPL that arithmetically simulates the truth values obtained by φ albeit in a smaller interval. A summary of our fuzzy logic characterisations is given in Table 1. In Appendix H, we discuss why no analogous characterisation can be given via infinite-valued Łukasiewicz logic.

Related work

Past fuzzy logic characterisations of neural networks have been obtained via representation theorems, showing that a given extension of Łukasiewicz logic is capable of expressing exactly all McNaughton functions with appropriate restrictions placed on the coefficients. For example, a well-known result by McNaughton [8] states that plain Łukasiewicz logic can express exactly all McNaughton functions with integer coefficients, linking Łukasiewicz logic to neural networks that use the truncated identity $f(x) = \min\{1, \max\{0, x\}\}$ (also known as the truncated ReLU) as the activation function. Amato, Di Nola and Gerla [9] gave a fuzzy logic characterisation for rational-parameter neural networks via Rational Łukasiewicz logic via McNaughton functions with rational coefficients. An analogous result was given by Di Nola, Gerla and Leustean in [10], connecting Łukasiewicz logic extended with multiplication with reals and neural networks of the above kind with real weights and biases via a representation theorem relating to McNaughton functions with real coefficients. Unlike the present paper, the activation values of the neural networks in the above characterisations, inputs and outputs included, were in the range $[0, 1]$. The activation function was the truncated identity function, whereas the present paper uses the ordinary ReLU-function. Rational Łukasiewicz logic used in [9], while similar to $\mathrm{RPL}(\odot)_{\leq 1}$ and $L\Pi\frac{1}{2}(\to_P^-)_{\leq 1}$, is more expressive than RPL and $L\Pi\frac{1}{2}(\odot^-, \to_P^-)$. Moreover, the above articles did not consider $\mathbb{Q}[\mathrm{ReLU}, x_i]_{i\in\mathbb{N}}$ or any analogous generalisation of neural networks.

There also exists a characterisation of neural networks via fuzzy logic that is not quite a straightforward logical characterisation. Castro and Trillas [11] gave a characterisation of neural networks that use real parameters and apply a squashing function on every layer via linear combinations of squashed transformations of propositions of Łukasiewicz logic.

In contrast to fuzzy logic characterisations, non-fuzzy logical characterisations of neural networks trace back to McCulloch and Pitts [12], who characterised neural networks with Boolean activation values and threshold activation functions via temporal propositional expressions. Much more recently, Ahvonen, Heiman and Kuusisto [13] gave a logical characterisation of recurrent neural networks operating with floating-point numbers via a recursive propositional logic called Boolean Network Logic (BNL). The same authors also gave an analogous characterisation of feedforward neural networks in [14]. Whereas the activation values, inputs and outputs included, are Boolean in [12] and floating-point numbers in [13, 14], and whereas the truth values in the characterising logics are Boolean, in the current paper all activation values are real numbers, and likewise the truth values of the characterising fuzzy logics are

reals from the closed interval $[0, 1]$.

Regarding connections between $L\Pi\frac{1}{2}$ and real algebra, Marchioni and Montagna [15] gave a constructive translation from terms of the language of ordered fields to terms of $L\Pi\frac{1}{2}$-algebra. This bears a close similarity to Lemma E.1 of the present paper, where we give a constructive translation from terms of $\mathbb{Q}[\mathrm{ReLU}, x_i]_{i\in\mathbb{N}}$ to formulae of the logic $L\Pi\frac{1}{2}(\to_P^-)$. As in the current paper, the correspondence between terms and their translations was shown by the semantics of the field of reals and the canonical semantics of $L\Pi\frac{1}{2}$-algebras over the interval $[0, 1]$. The translation in [15] is uniform, based on a single non-linear function that bijectively maps the open interval $]0, 1[$ to $\mathbb{R}$, and it was used to further interpret the universal theory of the reals in the equational theory of $L\Pi\frac{1}{2}$-algebras. By contrast, the translation in the current paper is based on a linear function, but it is non-uniform; for each positive integer k, we use a separate linear function that maps the interval $[-k, k]$ to the interval $[0, 1]$. In Lemma E.2, we provide a reverse translation from $L\Pi\frac{1}{2}(\to_P^-)$ to $\mathbb{Q}[\mathrm{ReLU}, x_i]_{i\in\mathbb{N}}$, but no reverse translation from terms of $L\Pi\frac{1}{2}$-algebra to terms of the language of ordered fields was given in [15]. Also, $\mathbb{Q}[\mathrm{ReLU}, x_i]_{i\in\mathbb{N}}$ is more expressive than the language of ordered fields as it also includes rational constants and the function ReLU, and likewise, our logic $L\Pi\frac{1}{2}(\to_P^-)$ is less expressive than the logic $L\Pi\frac{1}{2}$ or the corresponding $L\Pi\frac{1}{2}$-algebra used in [15], i.e., we can express a wider variety of terms in a weaker logic.

Recently, there have been many logical characterisations of graph neural networks (GNNs). In [16], Barceló et al. characterised GNNs in restriction to first-order logic (FO) via graded modal logic. They also showed that every property expressible by a formula of two-variable FO with counting is expressible by a GNN with global readout. Recently in [17], Hauke and Wałęga showed that the reverse does not hold. In the wake of [16], there have been numerous other (non-fuzzy) logical characterisations of GNNs [18, 19, 20, 21, 22, 23, 24] and of recurrent GNNs [25, 26, 27], and the lists sampled here are still growing.

There are various studies extending fuzzy logics with truth constants that generalise the ordinary truth constants $\top$ ("always true") and $\bot$ ("always false") of Boolean logics. While the Łukasiewicz implication $\varphi \to_L \psi$ is valid exactly when the truth value of ψ is always at least as great as the truth value of φ (see Section 2.2.1), the inclusion of truth constants $\overline{r}$ for all $r \in S$ for some $S \subseteq [0, 1]$ allows the truth degrees of formulae to be studied independently. In particular, it allows formulae $\overline{r} \to_L \psi$ which are valid exactly when the truth value of ψ is always at least as great as r. This allows generalising the notions of truth and provability of formulae to cover various degrees of truth and provability. Pavelka-style completeness is when these generalised notions of truth and provability coincide, named after Jan Pavelka who showed in [3, 4, 5] that Łukasiewicz logic extended with a truth constant for each real in $[0, 1]$, called Pavelka Logic, enjoys this property. While the number of truth constants makes the language of Pavelka Logic nondenumerable, Hájek showed in [2] that Rational Pavelka Logic, obtained from Pavelka Logic by including truth constants only for rational numbers in $[0, 1]$, also enjoys Pavelka-style completeness. In [6], Hájek also showed Pavelka-style completeness for the logic $\mathrm{RPL}(\odot)$ obtained from Rational Pavelka Logic by adding the product conjunction. Extensions of other fuzzy logics with truth constants have been studied, e.g., [28, 29, 30, 7, 31]. So far, truth constants do not appear to have a status other than as a technical means to calibrate degrees of truth., i.e., they do not appear to hold any independent status. By contrast, truth constants play a central role in the present paper, and our results establish a natural connection between truth constants and the weights and biases of neural networks.

2. Preliminaries

Let $\mathbb{N}$ denote the set of non-negative integers, $\mathbb{Z}$ the set of integers, $\mathbb{Q}$ the set of rational numbers and $\mathbb{R}$ the set of real numbers. We let $\mathbb{N}_+$, $\mathbb{Z}_+$, $\mathbb{Q}_+$ and $\mathbb{R}_+$ denote the restrictions of $\mathbb{N}$, $\mathbb{Z}$, $\mathbb{Q}$ and $\mathbb{R}$ to positive numbers. Given $a, b \in \mathbb{R}$ such that $a \leq b$, we let $[a, b]$ denote the closed interval $\{\, x \in \mathbb{R} \mid a \leq x \leq b \,\}$. Likewise, we define the open interval $]a, b[:= [a, b] \setminus \{a, b\}$ and $[0, \infty[:= \{\, x \in \mathbb{R} \mid x \geq 0 \,\}$.

Let $\mathrm{PROP} = \{ p_i \mid i \in \mathbb{N} \}$ be a countably infinite set of proposition symbols $p_0, p_1, \ldots$ and let $\mathrm{VAR} = \{ x_i \mid i \in \mathbb{N} \}$ be a countably infinite set of variables $x_0, x_1, \ldots$. We may use metasymbols $p, q, r, \ldots$ to denote proposition symbols in PROP and $x, y, z, \ldots$ to denote variables in VAR. We fix a bijection $b : \mathrm{PROP} \to \mathrm{VAR}$, and for each $p \in \mathrm{PROP}$ we let x_p denote $b(p)$, and for each $x \in \mathrm{VAR}$ we let p_x denote $b^{-1}(x)$.

A **valuation** is a function $V\colon \mathrm{PROP} \to [0,1]$ assigning a value $V(p) \in [0,1]$ to each proposition symbol $p \in \mathrm{PROP}$. Intuitively, $V(p)$ is the "degree of truth" of the proposition symbol p. An **evaluation map** is a function $E\colon \mathrm{VAR} \to \mathbb{R}$ that assigns a real value $E(x) \in \mathbb{R}$ to each variable $x \in \mathrm{VAR}$.

2.1. Neural networks and ReLU-polynomials

In this section, we introduce neural networks and neurons as well as the generalised polynomial ring $\mathbb{Q}[\mathrm{ReLU}, x_i]_{i\in\mathbb{N}}$ that subsumes neurons.

2.1.1. $\mathbb{Q}[\mathrm{ReLU}, x_i]_{i\in\mathbb{N}}$

We begin by defining a class of generalised polynomials that our neurons will be embedded in. It is based on the polynomial ring $\mathbb{Q}[x_i]_{i\in\mathbb{N}}$, i.e., the polynomial ring over $\mathbb{Q}$ with countably many variables. We do not require these polynomials to be in a reduced form; for example, we consider both $x(y+z)$ and $xy+xz$ to be valid polynomial expressions. We only make one modification to $\mathbb{Q}[x_i]_{i\in\mathbb{N}}$: we add the function $\mathrm{ReLU} : \mathbb{R} \to [0,\infty[$, $\mathrm{ReLU}(x) = \max\{0, x\}$ as another building block. We call such modified polynomials ReLU-*polynomials* or *terms*, and the structure consisting of them $\mathbb{Q}[\mathrm{ReLU}, x_i]_{i\in\mathbb{N}}$.

More formally, ReLU-**polynomials** or **terms of** $\mathbb{Q}[\mathrm{ReLU}, x_i]_{i\in\mathbb{N}}$ are constructed recursively as follows:

- For each $r \in \mathbb{Q}$, $\overline{r}$ is a term.
- Each $x \in \mathrm{VAR}$ is a term.
- If t and t' are terms, then $(t + t')$ and $(t \cdot t')$ are terms.
- If t is a term, then $\mathrm{ReLU}(t)$ is a term.

When there is no danger of confusion, we omit the parentheses from $(t + t')$ and $(t \cdot t')$. For brevity, we will usually write tt' to denote $t \cdot t'$. We will also write $-t$ to denote $\overline{-1} \cdot t$ and $t - t'$ to denote $t + (-t')$.

We interpret terms of $\mathbb{Q}[\mathrm{ReLU}, x_i]_{i\in\mathbb{N}}$ over the field of real numbers as follows. An evaluation map E fixes a value for each term when $+$, $\cdot$ and ReLU are given natural interpretations over the field of reals.

- If $t = \overline{r}$ for some $r \in \mathbb{Q}$, then $E(t) = r$.
- If $t = t' + t''$ for some terms t' and t'', then $E(t) = E(t') + E(t'')$.[1]
- If $t = t' \cdot t''$ for some terms t' and t'', then $E(t) = E(t')E(t'')$.
- If $t = \mathrm{ReLU}(t')$ for some term t', then $E(t) = \max\{0, E(t')\}$.

For each $k \in \mathbb{Z}_+$ and terms $t_1, \ldots, t_k$, we write $\sum_{i=1}^{k} t_i$ to denote the term $t_1 + \cdots + t_n$. The omitted parentheses in the sum can be placed in some canonical way, but given the semantics above, $+$ is associative and thus it does not make a difference how the parentheses are placed.

[1] Here and throughout the paper, we abuse notation and use $+$ and $\cdot$ to denote both a symbol in the syntax of $\mathbb{Q}[\mathrm{ReLU}, x_i]_{i\in\mathbb{N}}$ and the standard sum and multiplication over the field of real numbers. We could use different symbols like $\boxplus$ and $\boxdot$ in the syntax to differentiate the syntax and the semantics, but this would be more likely to increase confusion than decrease it. In the case of rationals however, it is useful to make the difference between syntax and semantics explicit, because a rational number can be represented in infinitely many ways, all of which can be covered with a single constant symbol.

Example 2.1. Let $t := \mathrm{ReLU}(x + xy) + y$. Now t is a term of $\mathbb{Q}[\mathrm{ReLU}, x_i]_{i\in\mathbb{N}}$, and given an evaluation map E such that $E(x) = 3$ and $E(y) = 0.5$, we have

$$E(t) = \max\{0, 3 + 3 \cdot 0.5\} + 0.5 = \max\{0, 4.5\} + 0.5 = 4.5 + 0.5 = 5.$$

Next, we define the degree of a term. Intuitively, degree relates to the maximum number of times that a variable is multiplied with another variable.

More formally, the **degree** of a term t, denoted $\deg(t)$, is defined recursively as follows:

- If $t = \overline{r}$ for some $r \in \mathbb{Q}$, then $\deg(t) = 0$.
- If $t = x$ for some $x \in \mathrm{VAR}$, then $\deg(t) = 1$.
- If $t = t' + t''$ for some terms t' and t'', then $\deg(t) = \max\{\deg(t'), \deg(t'')\}$.
- If $t = t't''$ for some terms t' and t'', then $\deg(t) = \deg(t') + \deg(t'')$.
- If $t = \mathrm{ReLU}(t')$ for some term t', then $\deg(t) = \deg(t')$.

When we discuss $\mathbb{Q}[\mathrm{ReLU}, x_i]_{i\in\mathbb{N}}$ **with degree bound** $d \in \mathbb{N}$, we mean the collection of exactly those terms t of $\mathbb{Q}[\mathrm{ReLU}, x_i]_{i\in\mathbb{N}}$ for which $\deg(t) \leq d$.

2.1.2. Neural networks

Next, we discuss neural networks. We begin by defining an auxiliary notion of a proto-neuron, which we will use throughout the paper. Then we define neural networks and neurons. Proto-neurons resemble neurons, but offer a more flexible syntax to work with, which is useful when constructing translations.

First, we consider proto-neurons, which are intuitively terms of $\mathbb{Q}[\mathrm{ReLU}, x_i]_{i\in\mathbb{N}}$ of degree at most 1. Syntactically, **proto-neurons** are constructed recursively as follows:

- For each $r \in \mathbb{Q}$, $\overline{r}$ is a proto-neuron.
- Each $x \in \mathrm{VAR}$ is a proto-neuron.
- If t and t' are proto-neurons, then $t + t'$ is a proto-neuron.
- If t and t' are proto-neurons such that $\deg(t) + \deg(t') \leq 1$, then tt' is a proto-neuron.
- If t is a proto-neuron, then $\mathrm{ReLU}(t)$ is a proto-neuron.

The following elucidating lemma characterises proto-neurons by their degree.

Lemma 2.2. *For all terms t of $\mathbb{Q}[\mathrm{ReLU}, x_i]_{i\in\mathbb{N}}$, t is a proto-neuron if and only if $\deg(t) \leq 1$.*

Proof. (Sketch) We prove the claim by induction over the structure of t; the details are in Appendix A. □

Example 2.3. The term $3(x + y)$ is a proto-neuron, but the term $(x + 3)y$ is not.

Proto-neurons, as the name suggests, resemble the neurons of a neural network. However, neurons only constitute a subclass of proto-neurons. Next, we introduce the concept of a neuron, which is closely related to the architecture of neural networks, which we shall define simultaneously. For an intuitive description of neural networks and neurons, see Section 1.

A **rational-parameter** ReLU**-activated neural network of depth** $d \in \mathbb{N}$ is defined recursively as follows. A neural network of depth 0 is a tuple $(y_1, \ldots, y_k)$ where $k \in \mathbb{Z}_+$ and $y_1, \ldots, y_k$ are distinct

variables in VAR. Next, assume we have defined neural networks of depth d. A neural network of depth $d+1$ is a tuple $N = (n_1, \ldots, n_k)$ where $k \in \mathbb{Z}_+$ and each n_j is a proto-neuron of the form $\mathrm{ReLU}\left(\sum_{i=1}^{\ell} \overline{w_{i,j}} n'_i + \overline{b_j}\right)$ where $\ell \in \mathbb{Z}_+$, $w_{i,j}, b_j \in \mathbb{Q}$ and $(n'_1, \ldots, n'_\ell)$ is a neural network of depth d. Note that ℓ and the neurons $n'_1, \ldots, n'_\ell$ are the same regardless of j. The components $n_1, \ldots, n_k$ of a neural network are called **neurons**; in particular, they are the **output neurons of** N. Each $\overline{w_{i,j}}$ is called the **weight** from the neuron n'_i to the neuron n_j,[2] and each $\overline{b_j}$ is called the **bias** of the neuron n_j.

Example 2.4. The proto-neuron $-x$ is not a neuron and neither is $\mathrm{ReLU}(\overline{2}x) + \mathrm{ReLU}(\overline{3}y)$, but the following proto-neuron is a neuron:

$$\mathrm{ReLU}\left(\overline{1} \cdot \mathrm{ReLU}(\overline{2}x + \overline{0}y + \overline{0}) + \overline{1} \cdot \mathrm{ReLU}(\overline{0}x + \overline{3}y + \overline{0}) + \overline{0}\right).$$

2.2. Logics

In this section, we introduce the variety of fuzzy logics that will be used to characterise $\mathbb{Q}[\mathrm{ReLU}, x_i]_{i\in\mathbb{N}}$ and neural networks. Key roles here are played by Rational Pavelka Logic RPL as well as another fuzzy logic called $L\Pi\frac{1}{2}$. Both of these are extensions of Łukasiewicz logic, which we will define first.

2.2.1. Łukasiewicz logic

The first logic we define is Łukasiewicz logic, which is syntactically analogous to propositional logic but interpreted differently. While the canonical semantics for propositional logic gives each formula a truth value 0 (false) or 1 (true), the canonical semantics for infinite-valued Łukasiewicz logic generalises this, giving each formula a truth value from the interval $[0, 1]$ intuitively allowing degrees of truth between true and false. Under this interpretation, certain arithmetic operations like sum and subtraction are definable, though truncated at the end points 0 and 1 of the interval.

The PROP**-formulae of infinite-valued Łukasiewicz logic** [1] (also called Łukasiewicz logic or the Łukasiewicz-Tarski logic) are defined by the following grammar:

$$\varphi ::= \overline{0} \mid p \mid \varphi \to_L \varphi,$$

where $p \in \mathrm{PROP}$. Intuitively, $\overline{0}$ is a formula that is always false, and $\varphi \to_L \psi$ is read "if φ, then ψ". Note that similar to terms of $\mathbb{Q}[\mathrm{ReLU}, x_i]_{i\in\mathbb{N}}$, logic formulae are also terms, only of a different language, but we call them formulae to keep the terminologies distinct from each other.

The semantics of infinite-valued Łukasiewicz logic is defined over MV-algebras [32, 33]. Rather than presenting the full variety of MV-algebras, we present the semantics given by the canonical MV-algebra defined over the real interval $[0, 1]$ which satisfies all the axioms of Łukasiewicz logic. Let $V : \mathrm{PROP} \to [0, 1]$ be a valuation. We extend the function V to determine the truth value of any formula of Łukasiewicz logic as follows:

- $V(\overline{0}) := 0$.
- $V(\varphi \to_L \psi) := \min\{1, 1 - V(\varphi) + V(\psi)\}$ for all formulae φ and ψ of Łukasiewicz logic.

For the reader more familiar with propositional logic, it may be useful to check how the connective $\to_L$ and subsequent connectives behave when $V(\varphi), V(\psi) \in \{0, 1\}$. In the case of $\to_L$, such an examination retrieves the canonical Boolean semantics for implication.

[2] Technically, there may be multiple such weights if some of the neurons $n'_1, \ldots, n'_\ell$ are identical to each other, since our definition of a neuron simply refers to an aggregation formula, but in such situations we naturally consider the identical neurons to be distinct from each other since they occupy separate positions in the neural network.

We define the following abbreviated connectives with the intuitive meaning given in parentheses: $\neg_L$ (negation), $\oplus$ (addition), $\otimes$ (addition with bias -1), $\ominus$ (subtraction), $\wedge$ (minimum), $\vee$ (maximum) and $\leftrightarrow_L$ (equivalence). The connectives $\neg_L$, $\oplus$ and $\ominus$ will be crucial later on, whereas the others may be of independent interest to the reader. We give a definition for each connective on the left and the resulting truth value of the formula in a valuation V on the right (the details are left to the reader):

$$\begin{aligned}
&\neg_L\varphi := \varphi \to_L \overline{0}, && V(\neg_L\varphi) = 1 - V(\varphi),\\
&\varphi \oplus \psi := \neg_L\varphi \to_L \psi, && V(\varphi \oplus \psi) = \min\{1, V(\varphi) + V(\psi)\},\\
&\varphi \otimes \psi := \neg_L(\varphi \to_L \neg_L\psi), && V(\varphi \otimes \psi) = \max\{0, V(\varphi) + V(\psi) - 1\},\\
&\varphi \ominus \psi := \varphi \otimes \neg_L\psi, && V(\varphi \ominus \psi) = \max\{0, V(\varphi) - V(\psi)\},\\
&\varphi \wedge \psi := \varphi \otimes (\varphi \to_L \psi), && V(\varphi \wedge \psi) = \min\{V(\varphi), V(\psi)\},\\
&\varphi \vee \psi := (\varphi \to_L \psi) \to_L \psi, && V(\varphi \vee \psi) = \max\{V(\varphi), V(\psi)\},\\
&\varphi \leftrightarrow_L \psi := (\varphi \to_L \psi) \wedge (\psi \to_L \varphi), && V(\varphi \leftrightarrow_L \psi) = 1 - |V(\varphi) - V(\psi)|.
\end{aligned}$$

These connectives are naturally definable in the same way in all extensions of Łukasiewicz logic.

2.2.2. Rational Pavelka Logic

Next, we define Rational Pavelka Logic (RPL), which extends Łukasiewicz logic by adding a truth constant for each rational number in the interval $[0, 1]$. Pavelka logic was originally introduced in [3, 4, 5], where a truth constant was included for each real number in the interval $[0, 1]$ including for irrational numbers. Later in [2] the rational version of Pavelka Logic was introduced, where truth constants are only included for rational numbers.

The PROP-**formulae of Rational Pavelka Logic (**RPL**)** are defined by the following grammar:

$$\varphi ::= \overline{r} \mid p \mid \varphi \to_L \varphi,$$

where $r \in \mathbb{Q} \cap [0, 1]$ and $p \in \mathrm{PROP}$. Intuitively, $\overline{r}$ is a formula that always has the truth value r.

The canonical semantics of RPL over the real interval $[0, 1]$ extends the canonical semantics of Łukasiewicz logic as follows. Let $V : \mathrm{PROP} \to [0, 1]$ be a valuation. The semantics for proposition symbols and $\to_L$ is defined as for Łukasiewicz logic. For truth constants, we define

- $V(\overline{r}) := r$ for each $r \in \mathbb{Q} \cap [0, 1]$.

2.2.3. RPL($\odot$)

In [6], Hájek introduced the logic RPL($\odot$), which extends RPL with the product conjunction $\odot$.[3] As the name suggests, the product conjunction allows multiplying the truth value of a formula with the truth value of another formula.

The PROP-**formulae of** RPL($\odot$) are defined by the following grammar:

$$\varphi ::= \overline{r} \mid p \mid \varphi \to_L \varphi \mid \varphi \odot \varphi,$$

where $r \in \mathbb{Q} \cap [0, 1]$ and $p \in \mathrm{PROP}$. Intuitively, $\odot$ corresponds to multiplication.

The canonical semantics of RPL($\odot$) over the real interval $[0, 1]$ extends the semantics of RPL as follows. Let $V : \mathrm{PROP} \to [0, 1]$ be a valuation. The semantics for truth constants, proposition symbols and $\to_L$ is defined as in RPL. For $\odot$, we define

- $V(\varphi \odot \psi) := V(\varphi)V(\psi)$ for all formulae φ and ψ of RPL($\odot$).

[3] In the literature on MV-algebras, the connective $\odot$ commonly refers to the dual operator of $\oplus$ denoted in this paper by $\otimes$. In this work, $\odot$ denotes a primitive connective that is not derived from $\oplus$ or $\to_L$.

2.2.4. $L\Pi\frac{1}{2}$

The logic $L\Pi\frac{1}{2}$ is another extension of Łukasiewicz logic. Rather than adding a truth constant for each rational number in $[0,1]$ like RPL and $\mathrm{RPL}(\odot)$, $L\Pi\frac{1}{2}$ only adds one for $\frac{1}{2}$. On top of the product conjunction $\odot$, $L\Pi\frac{1}{2}$ adds the corresponding residual division connective: the product implication $\to_P$.

The PROP-**formulae of** $L\Pi\frac{1}{2}$ are obtained by the following grammar:

$$\varphi ::= \overline{0} \mid \overline{\tfrac{1}{2}} \mid p \mid \varphi \to_L \varphi \mid \varphi \odot \varphi \mid \varphi \to_P \varphi,$$

where $p \in \mathrm{PROP}$. Intuitively, $\to_P$ corresponds to division.

The canonical semantics of $L\Pi\frac{1}{2}$ over $[0,1]$ is defined as follows. Let $V : \mathrm{PROP} \to [0,1]$ be a valuation. The semantics for truth constants, proposition symbols, $\to_L$ and $\odot$ is defined as in $\mathrm{RPL}(\odot)$. The semantics of $\to_P$ is defined as follows. Given formulae φ and ψ of $L\Pi\frac{1}{2}$,

$$V(\varphi \to_P \psi) := \begin{cases} 1, & \text{if } V(\varphi) \leq V(\psi) \\ \frac{V(\psi)}{V(\varphi)}, & \text{otherwise.} \end{cases}$$

Note that $V(\varphi) = 0$ implies $V(\varphi) \leq V(\psi)$. Thus, the denominator of $\frac{V(\psi)}{V(\varphi)}$ never goes to zero.

It is shown in [7] that for each $r \in \mathbb{Q} \cap [0,1]$, we can construct a formula $\underline{r}$ of $L\Pi\frac{1}{2}$ without proposition symbols such that $V(\underline{r}) = r$ for each valuation V. We show an alternative construction in Appendix B.

2.2.5. Fragments of $\mathrm{RPL}(\odot)$ and $L\Pi\frac{1}{2}$

In this section, we define various syntactic fragments of $L\Pi\frac{1}{2}$ and $\mathrm{RPL}(\odot)$. We first establish two fragments of $L\Pi\frac{1}{2}$ that place limitations on the use of the connectives $\odot$ and $\to_P$. Then, we define a hierarchy of fragments of $L\Pi\frac{1}{2}$ and $\mathrm{RPL}(\odot)$, starting with fragments containing no proposition symbols and fragment-by-fragment allowing more proposition symbols nested within product conjunctions $\odot$.

We start with an auxiliary fragment of $L\Pi\frac{1}{2}$ that consists of the formulae of $L\Pi\frac{1}{2}$ that do not contain proposition symbols. This means that the truth values of formulae do not depend on the valuation. More formally, the **formulae of** $L\Pi\frac{1}{2}{}_{\leq 0}$ are constructed according to the following grammar:

$$\varphi ::= \overline{0} \mid \overline{\tfrac{1}{2}} \mid \varphi \to_L \varphi \mid \varphi \odot \varphi \mid \varphi \to_P \varphi.$$

Next, we define a fragment of $L\Pi\frac{1}{2}$ where proposition symbols do not appear in the scope of the connectives $\odot$ and $\to_P$. This means the truth values of multiplications and divisions do not depend on the valuation. The PROP-**formulae of** $L\Pi\frac{1}{2}(\odot^-, \to_P^-)$ are constructed according to the grammar:

$$\varphi ::= \psi \mid p \mid \varphi \to_L \varphi,$$

where ψ is a formula of $L\Pi\frac{1}{2}{}_{\leq 0}$ and $p \in \mathrm{PROP}$.

Analogously, we define a fragment of $L\Pi\frac{1}{2}$ that only restricts the use of $\to_P$, i.e., only the truth values of divisions do not depend on the valuation. The PROP-**formulae of** $L\Pi\frac{1}{2}(\to_P^-)$ are constructed according to the following grammar:

$$\varphi ::= \psi \mid p \mid \varphi \to_L \varphi \mid \varphi \odot \varphi,$$

where ψ is a formula of $L\Pi\frac{1}{2}{}_{\leq 0}$ and $p \in \mathrm{PROP}$.

Next, we define a hierarchy of syntactic fragments of both $\mathrm{RPL}(\odot)$ and $L\Pi\frac{1}{2}(\to_P^-)$. First, we define the fragments $\mathrm{RPL}(\odot)_{\leq 0}$ and $L\Pi\frac{1}{2}(\to_P^-)_{\leq 0}$ which do not contain proposition symbols. In the case of $L\Pi\frac{1}{2}(\to_P^-)$, the **formulae of** $L\Pi\frac{1}{2}(\to_P^-)_{\leq 0}$ are simply those of the logic $L\Pi\frac{1}{2}{}_{\leq 0}$ defined above. The **formulae of** $\mathrm{RPL}(\odot)_{\leq 0}$ are obtained according to the following grammar:

$$\varphi ::= \overline{r} \,|\, \varphi \to_L \varphi \,|\, \varphi \odot \varphi,$$

where $r \in \mathbb{Q} \cap [0,1]$.

Next, let $\mathcal{L}$ be either $\mathrm{RPL}(\odot)$ or $L\Pi\frac{1}{2}(\to_P^-)$ and assume we have defined the fragment $\mathcal{L}_{\leq n}$ where $n \in \mathbb{N}$. The PROP**-formulae of** $\mathcal{L}_{\leq n+1}$ are obtained according to the following grammar:

$$\varphi ::= p \,|\, \psi \,|\, \alpha \odot \beta \,|\, \varphi \to_L \varphi,$$

where $p \in \mathrm{PROP}$, ψ is a formula of $\mathcal{L}_{\leq n}$, and α and β are formulae of $\mathcal{L}_{\leq i}$ and $\mathcal{L}_{\leq j}$ respectively for any $i, j \in \mathbb{N}$ such that $i + j \leq n + 1$. If $n \geq 1$, then the atomic proposition symbols p could be omitted from the grammar, as they would be implicitly included with ψ. The fragments $\mathcal{L}_{\leq 0}$, $\mathcal{L}_{\leq 1}$, ... form a hierarchy where each fragment is subsumed by the next.

3. Equivalence

In this section, we define various notions of equivalence between terms of $\mathbb{Q}[\mathrm{ReLU}, x_i]_{i\in\mathbb{N}}$ and/or formulae of logics. The most basic notions are centred around formulae and/or terms receiving identical values with matching valuations and evaluation maps. A more involved notion of equivalence involves scaling the values in real intervals down to the interval $[0,1]$, since unlike terms of $\mathbb{Q}[\mathrm{ReLU}, x_i]_{i\in\mathbb{N}}$, the truth values of formulae of $\mathrm{RPL}(\odot)$ and $L\Pi\frac{1}{2}$ are always in $[0,1]$.

3.1. Non-scaled equivalence

In this section, we consider notions of equivalence based on receiving identical values with identical inputs. We start by defining equivalence between terms.

Definition 3.1. We say that terms t and t' of $\mathbb{Q}[\mathrm{ReLU}, x_i]_{i\in\mathbb{N}}$ are **equivalent** if $E(t) = E(t')$ for each evaluation map E. For two tuples of terms $T = (t_1, \ldots, t_k)$ and $T' = (t'_1, \ldots, t'_k)$ of equal length, we say that T and T' are **equivalent** if the terms t_i and t'_i are equivalent for each $i \in \{1, \ldots, k\}$.

Example 3.2. The terms $x + y$ and $\frac{\overline{1}}{2}x + \frac{\overline{1}}{2}x + \overline{2}y - y$ are equivalent. The terms x and $\mathrm{ReLU}(x)$ are not: if E is an evaluation map such that $E(x) < 0$, then we get $E(\mathrm{ReLU}(x)) = \max\{0, E(x)\} = 0 \neq E(x)$.

Next, we establish equivalence between formulae of logics.

Definition 3.3. We say that two formulae φ and ψ of $\mathrm{RPL}(\odot)$ or $L\Pi\frac{1}{2}$ are **equivalent** if $V(\varphi) = V(\psi)$ for each valuation V.

Example 3.4. The formulae $\frac{\overline{1}}{4}$ and $\frac{\overline{1}}{2} \odot \frac{\overline{1}}{2}$ are equivalent. So are the formulae φ and $\neg_L\neg_L\varphi$.

Lastly, we consider normal equivalence between terms of $\mathbb{Q}[\mathrm{ReLU}, x_i]_{i\in\mathbb{N}}$ and formulae of logics.

Definition 3.5. We say that a term t of $\mathbb{Q}[\mathrm{ReLU}, x_i]_{i\in\mathbb{N}}$ and a formula φ of $\mathrm{RPL}(\odot)$ or $L\Pi\frac{1}{2}$ are **equivalent** if given any evaluation map E and valuation V such that $E(x) = V(p_x)$ for all $x \in \mathrm{VAR}$:

$$E(t) = V(\varphi).$$

3.2. Connecting fragments of $\mathrm{RPL}(\odot)$ and $L\Pi\frac{1}{2}$

In this section, we establish a non-scaled notion of expressive power between logics and use it to match fragments of $\mathrm{RPL}(\odot)$ and $L\Pi\frac{1}{2}$.

Definition 3.6. We say that two logics $\mathcal{L}$ and $\mathcal{L}'$ **have the same expressive power** if for each formula of $\mathcal{L}$ there is an equivalent formula of $\mathcal{L}'$ and vice versa.

We next establish natural connections between fragments of $\mathrm{RPL}(\odot)$ and $L\Pi\frac{1}{2}$.

Proposition 3.7. *For all $n \in \mathbb{N}$, the logics $\mathrm{RPL}(\odot)_{\leq n}$ and $L\Pi\frac{1}{2}(\to^-_P)_{\leq n}$ have the same expressive power.*

Proof. (Sketch) We show the claim by induction over n. The details are in Appendix C. □

By similar reasoning, we also obtain the following two propositions.

Proposition 3.8. $\mathrm{RPL}(\odot)$ *and $L\Pi\frac{1}{2}(\to^-_P)$ have the same expressive power.*

Proposition 3.9. RPL *and $L\Pi\frac{1}{2}(\odot^-, \to^-_P)$ have the same expressive power.*

3.3. Scaled equivalence

In this section we define scaled notions of equivalence. First, we define a necessary scaling function.

Definition 3.10. Let $k \in \mathbb{R}_+$. We define the bijection $\mathrm{scale}_k : \mathbb{R} \to \mathbb{R}$, $\mathrm{scale}_k(x) = \frac{k+x}{2k}$.

Clearly, scale_k scales the interval $[-k, k]$ to the interval $[0, 1]$, as we have $\mathrm{scale}_k(-k) = 0, \mathrm{scale}_k(0) = \frac{1}{2}$ and $\mathrm{scale}_k(k) = 1$. The idea behind the following notion of equivalence is that, the truth values of formulae being restricted to $[0, 1]$, a formula can simulate a term when both the inputs and outputs are scaled down using scale_k. This naturally means that the term can only gain values within the interval $[-k, k]$, and to accomplish this, the input interval also naturally needs to be restricted, as otherwise even a simple term such as x could obtain arbitrarily great values.

Definition 3.11. Let $i, k \in \mathbb{R}_+$ such that $i \leq k$. Let φ be a formula of $\mathrm{RPL}(\odot)$ or $L\Pi\frac{1}{2}$ and let t be a term of $\mathbb{Q}[\mathrm{ReLU}, x_i]_{i\in\mathbb{N}}$. We say that φ is (i, k)-**equivalent** to t if given any evaluation map E such that $E(x) \in [-i, i]$ for each $x \in \mathrm{VAR}$ and valuation V such that $V(p) = \mathrm{scale}_k(E(x_p))$ for all $p \in \mathrm{PROP}$, we have

$$V(\varphi) = \mathrm{scale}_k(E(t)).$$

Lastly, we use the scaling function scale_k to define an alternative notion of equivalence between formulae of logics. This equivalence notion applies scale_k between two logic formulae. Intuitively, this means that a formula simulates another formula, but does so in the interval $[\mathrm{scale}_k(0), \mathrm{scale}_k(1)] = \left[\frac{1}{2}, \frac{k+1}{2k}\right]$.

Definition 3.12. Let $k \in \mathbb{R}_+$, let φ be a formula of $\mathrm{RPL}(\odot)$ or $L\Pi\frac{1}{2}$ and likewise for ψ. We say that ψ is k-**equivalent** to φ if given any valuations V and V' such that $V'(p) = \mathrm{scale}_k(V(p))$ for each $p \in \mathrm{PROP}$, we have

$$V'(\psi) = \mathrm{scale}_k(V(\varphi)).$$

3.4. Scaled expressive power

In this section, we define scaled notions of expressive power between $\mathbb{Q}[\mathrm{ReLU}, x_i]_{i\in\mathbb{N}}$ and/or logics. We start with a scaled notion of expressive power between logics.

Definition 3.13. Let $\mathcal{L}$ and $\mathcal{L}'$ be logics. We say that $\mathcal{L}$ has **the same expressive power as $\mathcal{L}'$ (w.r.t. scaling)** if the following conditions hold:

1. for each formula of $\mathcal{L}$, there exists a k-equivalent formula of $\mathcal{L}'$ for some $k \in \mathbb{R}_+$, and
2. for each formula of $\mathcal{L}'$, there exists an equivalent formula of $\mathcal{L}$.

Lastly, we establish a scaled notion of expressive power that we use in our characterisation theorems.

Definition 3.14. Let $\mathcal{T}$ be a subclass of $\mathbb{Q}[\mathrm{ReLU}, x_i]_{i\in\mathbb{N}}$ (or of tuples thereof) and let $\mathcal{L}$ be one of the defined logics. We say that $\mathcal{T}$ and $\mathcal{L}$ have **the same expressive power (w.r.t. scaling)** if for each $n \in \mathbb{Z}_+$ the following conditions hold:

1. for each tuple $(t_1, \ldots, t_n)$ of $\mathcal{T}$ and all $i \in \mathbb{R}_+$, there exists some $k \geq i$ and a tuple $(\varphi_1, \ldots, \varphi_n)$ of formulae of $\mathcal{L}$ such that φ_j is (i, k)-equivalent to t_j for each $j \in \{1, \ldots, n\}$, and
2. for each tuple $(\varphi_1, \ldots, \varphi_n)$ of formulae of $\mathcal{L}$ there exists a tuple $(t_1, \ldots, t_n)$ of $\mathcal{T}$ such that for each $j \in \{1, \ldots, n\}$ the term t_j and the formula φ_j are equivalent.

Note that although $\mathcal{T}$ can be either a subclass of terms or tuples of terms of $\mathbb{Q}[\mathrm{ReLU}, x_i]_{i\in\mathbb{N}}$, in both cases we translate tuples of terms to tuples of formulae and vice versa. Appropriately, the numbers i and k in the first condition must be independent of which components of the tuple are being compared. For an illustration of how the scaling works in this kind of characterisation, see Figure 2.

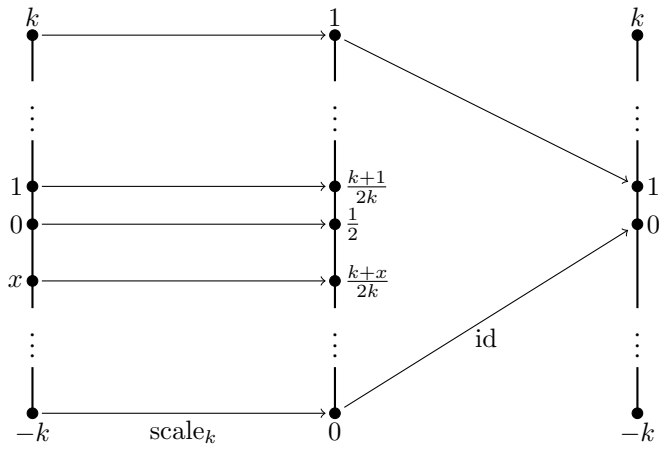

Figure 2: An illustration of the scaling in our translations. The left and right number lines correspond to values of terms, while the middle number line shows the truth values of formulae. From terms to formulae (left-to-middle), we scale $[-k, k]$ linearly into $[0, 1]$, while from formulae to terms (middle-to-right) we use the identity function.

4. Fuzzy logic characterisations

In this section, we give fuzzy logic characterisations of $\mathbb{Q}[\mathrm{ReLU}, x_i]_{i\in\mathbb{N}}$ and specifically of neural networks. In Section 4.1, we give a fuzzy logic characterisation of $\mathbb{Q}[\mathrm{ReLU}, x_i]_{i\in\mathbb{N}}$, as well as of

subclasses of $\mathbb{Q}[\mathrm{ReLU}, x_i]_{i\in\mathbb{N}}$ obtained by giving a constant upper bound for the degrees of terms. Then, we give a fuzzy logic characterisation of neural networks in Section 4.2.

4.1. Characterising $\mathbb{Q}[\mathrm{ReLU}, x_i]_{i\in\mathbb{N}}$

In this section, we give a fuzzy logic characterisation for $\mathbb{Q}[\mathrm{ReLU}, x_i]_{i\in\mathbb{N}}$ with and without degree bound via the logics $\mathrm{RPL}(\odot)$ and $L\Pi\frac{1}{2}(\to_P^-)$ and their fragments $\mathrm{RPL}(\odot)_{\leq n}$ and $L\Pi\frac{1}{2}(\to_P^-)_{\leq n}$. As a reminder, in our characterisations we translate tuples of terms to tuples of formulae and vice versa.

Theorem 4.1. *$\mathbb{Q}[\mathrm{ReLU}, x_i]_{i\in\mathbb{N}}$ has the same expressive power (w.r.t. scaling) as $\mathrm{RPL}(\odot)$ and $L\Pi\frac{1}{2}(\to_P^-)$.*

Proof. (Sketch) First, we define auxiliary connectives in $\mathrm{RPL}(\odot)$ and $L\Pi\frac{1}{2}(\to_P^-)$ that simulate the addition and multiplication of reals in a given interval $[-k, k]$ down in the interval $[0, 1]$; the details of this are given in Appendix D. We then prove the theorem by giving recursive translations from terms of $\mathbb{Q}[\mathrm{ReLU}, x_i]_{i\in\mathbb{N}}$ to formulae of $\mathrm{RPL}(\odot)$ and $L\Pi\frac{1}{2}(\to_P^-)$, and vice versa. The correctness of the translations is proven by induction. The details are given in Appendix E. □

By restricting the degree of the terms and the fragment of the logic in the above theorem and carrying the restriction through the proof, we obtain the following fuzzy logic characterisation of $\mathbb{Q}[\mathrm{ReLU}, x_i]_{i\in\mathbb{N}}$ with degree bound $d \in \mathbb{N}$ via the logics $\mathrm{RPL}(\odot)_{\leq d}$ and $L\Pi\frac{1}{2}(\to_P^-)_{\leq d}$.

Theorem 4.2. *For each $d \in \mathbb{N}$, $\mathbb{Q}[\mathrm{ReLU}, x_i]_{i\in\mathbb{N}}$ with degree bound d has the same expressive power (w.r.t. scaling) as $\mathrm{RPL}(\odot)_{\leq d}$ and $L\Pi\frac{1}{2}(\to_P^-)_{\leq d}$.*

So far in relation to both $\mathbb{Q}[\mathrm{ReLU}, x_i]_{i\in\mathbb{N}}$ and RPL, we have restricted to the case where constant parameters and truth constants are rational numbers. Analogously to $\mathbb{Q}[\mathrm{ReLU}, x_i]_{i\in\mathbb{N}}$ and RPL, we can define $\mathbb{R}[\mathrm{ReLU}, x_i]_{i\in\mathbb{N}}$ and Pavelka Logic (PL) by replacing the rationals in the definitions of $\mathbb{Q}[\mathrm{ReLU}, x_i]_{i\in\mathbb{N}}$ and RPL with reals. All the same notions and lemmas from Sections 2 and 3 and Appendices C and D relating to $\mathbb{Q}[\mathrm{ReLU}, x_i]_{i\in\mathbb{N}}$ and RPL can then analogously be defined and obtained for $\mathbb{R}[\mathrm{ReLU}, x_i]_{i\in\mathbb{N}}$ and PL, including real-parameter neural networks, the logic $\mathrm{PL}(\odot)$ and the fragments $\mathrm{PL}(\odot)_{\leq n}$ for each $n \in \mathbb{N}$. We also note that the proofs of Theorems 4.1 and 4.2 did not hinge upon the restriction to rationals. Thus, we obtain the following corollaries.

Corollary 4.3. *$\mathbb{R}[\mathrm{ReLU}, x_i]_{i\in\mathbb{N}}$ has the same expressive power (w.r.t. scaling) as $\mathrm{PL}(\odot)$.*

Corollary 4.4. *For each $d \in \mathbb{N}$, $\mathbb{R}[\mathrm{ReLU}, x_i]_{i\in\mathbb{N}}$ with degree bound d has the same expressive power (w.r.t. scaling) as $\mathrm{PL}(\odot)_{\leq d}$.*

We primarily considered $\mathbb{Q}[\mathrm{ReLU}, x_i]_{i\in\mathbb{N}}$ rather than $\mathbb{R}[\mathrm{ReLU}, x_i]_{i\in\mathbb{N}}$ because in the case of rationals we also obtain a characterisation via $L\Pi\frac{1}{2}(\to_P^-)$. With reals, this would require making the vocabulary of $L\Pi\frac{1}{2}(\to_P^-)$ nondenumerable by, e.g., adding a truth constant for each real number in $[0, 1]$, because even with a denumerable vocabulary the number of pairwise non-equivalent formulae would be only denumerable, i.e., not even enough to express every real in $[0, 1]$. However, the core appeal of $L\Pi\frac{1}{2}(\to_P^-)$ over $\mathrm{RPL}(\odot)$ is that the vocabulary of $L\Pi\frac{1}{2}(\to_P^-)$ is essentially finite whereas the vocabulary of $\mathrm{RPL}(\odot)$ is denumerable. Accordingly, the appeal of $\mathrm{RPL}(\odot)$ over $L\Pi\frac{1}{2}(\to_P^-)$ is that due to the richer vocabulary, a formula of $\mathrm{RPL}(\odot)$ is sometimes significantly shorter than the shortest equivalent formula of $L\Pi\frac{1}{2}(\to_P^-)$, and we also know from [6] that $\mathrm{RPL}(\odot)$ enjoys Pavelka-style completeness.

4.2. Characterising neural networks

In this section, we give fuzzy logic characterisations of neural networks.

Theorem 4.5. *Neural networks have the same expressive power (w.r.t. scaling) as the logics* $\mathrm{RPL}(\odot)_{\leq 1}$, $L\Pi\frac{1}{2}(\to_P^-)_{\leq 1}$, RPL *and* $L\Pi\frac{1}{2}(\odot^-, \to_P^-)$.

Proof. (Sketch) We first give a recursive constructive translation from formulae of $\mathrm{RPL}(\odot)_{\leq 1}$ and $L\Pi\frac{1}{2}(\to_P^-)_{\leq 1}$ into neurons. Then, we give a recursive constructive translation from proto-neurons to formulae of RPL and $L\Pi\frac{1}{2}(\odot^-, \to_P^-)$; this direction involves inflating the proto-neurons such that they only contain integers, after which multiplication by a constant can be simulated in the logic via addition. The correctness of the translation is shown via induction. The details are in Appendix F. □

Example 4.6. Consider the formula $\varphi = \overline{\frac{1}{3}} \to_L \left(p \odot \overline{\frac{1}{2}}\right)$ of $\mathrm{RPL}(\odot)_{\leq 1}$. The following neuron n_φ is equivalent to φ:

$$\mathrm{ReLU}\left(\overline{-1} \cdot \mathrm{ReLU}\left(\overline{1} \cdot \mathrm{ReLU}\left(\overline{0}x_p + \overline{\frac{1}{3}}\right) + \overline{-1} \cdot \mathrm{ReLU}\left(\overline{\frac{1}{2}}x_p + \overline{0}\right) + \overline{0}\right) + \overline{1}\right).$$

An illustration of the neural network $N := (n_\varphi)$ is given in Figure 3.

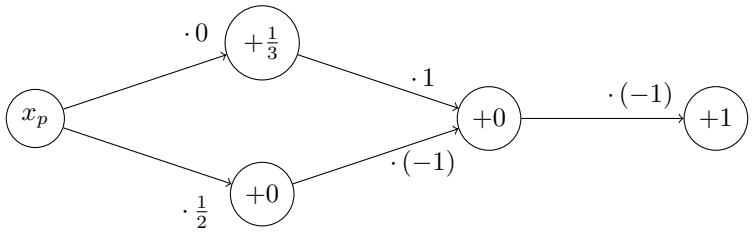

Figure 3: An illustration of the neural network N. Weights are written along edges and biases inside neurons.

Translating neurons into formulae requires the auxiliary connectives defined in Appendix D; for an example, see Appendix F.

In the proof of Theorem 4.5, the translation from $\mathrm{RPL}(\odot)_{\leq 1}$ to neurons does not hinge on the constant symbols in the neurons and the truth constants in the formulae being rationals; the same translation works with reals. The translation from proto-neurons to RPL does depend on this restriction, but if we only want to translate neurons into $\mathrm{RPL}(\odot)_{\leq 1}$ instead of RPL, then we can use Theorem 4.2 where the proof does not depend on restricting to rationals. Thus, we obtain the following corollary.

Corollary 4.7. *Real-parameter neural networks have the same expressive power (w.r.t. scaling) as* $\mathrm{PL}(\odot)_{\leq 1}$.

Again, the reason we focused on neural networks with rational weights and biases was because we were also able to obtain characterisations via RPL, $L\Pi\frac{1}{2}(\to_P^-)_{\leq 1}$ and $L\Pi\frac{1}{2}(\odot^-, \to_P^-)$. For reals, we could obtain a characterisation via adding a truth constant for each real in $[0, 1]$ to $L\Pi\frac{1}{2}(\to_P^-)_{\leq 1}$, whereas for RPL and $L\Pi\frac{1}{2}(\odot^-, \to_P^-)$ even this would not suffice to obtain a characterisation. The characterisation via RPL is especially interesting, since RPL is known to enjoy Pavelka-style completeness [2].

From Theorem 4.5, we obtain a corollary between $\mathrm{RPL}(\odot)_{\leq 1}$, $L\Pi\frac{1}{2}(\to_P^-)_{\leq 1}$, RPL and $L\Pi\frac{1}{2}(\odot^-, \to_P^-)$.

Corollary 4.8. $\mathrm{RPL}(\odot)_{\leq 1}$ *and* $L\Pi\frac{1}{2}(\to_P^-)_{\leq 1}$ *have the same expressive power (w.r.t. scaling) as* RPL *and* $L\Pi\frac{1}{2}(\odot^-, \to_P^-)$.

Proof. (Sketch) We first show that if a formula φ and a term t are equivalent and the formula ψ is (i,k)-equivalent to t for some $1 \leq i \leq k$, then ψ is k-equivalent to φ. The result then follows from the translations implied by Theorem 4.5. The details are in Appendix G. □

In Appendix H, we consider some proto-neurons that cannot be translated into (i,k)-equivalent formulae of Łukasiewicz logic. Conversely, we study what kinds of neurons are expressible in Łukasiewicz logic.

5. Conclusion

We have given fuzzy logic characterisations of ReLU-activated rational-weight neural networks via fragments of two fuzzy logics, $\mathrm{RPL}(\odot)$ and $L\Pi\frac{1}{2}$. We also gave a fuzzy logic characterisation of a more general structure consisting of polynomials from the polynomial ring over $\mathbb{Q}$ in countably many variables appended with ReLU. The inputs for the neural networks and for the polynomials are allowed to be arbitrary real numbers. Future work could involve studying how neural networks are connected to tautologies of the characterising fuzzy logics, as well as studying the completeness properties of the fuzzy logic fragments introduced here.

Logic-based approaches to AI systems are useful for, e.g., verification and explainability. Verification is important in applications with safety-related components. Explainability, on the other hand, concerns the human interface: it is precisely in such contexts that AI replacing humans may be unlikely, since human understanding itself serves as the primary point of reference. Moreover, explainability is also central in scientific uses of AI, where it is typically not sufficient to obtain results without understanding the reasons behind them. Explainability is thus clearly important and naturally motivates the study of logics that are cognitively easy to process. Whether fuzzy logic satisfies this requirement is debatable, but investigating different kinds of logics helps to clarify the overall landscape. Furthermore, logical methods can also provide new perspectives on training AI systems, potentially leading to useful algorithmic approaches that differ from standard gradient-descent methods.

Acknowledgments

Damian Heiman was supported by the Magnus Ehrnrooth Foundation. Antti Kuusisto was supported by the project *Perspectives on computational logic*, funded by the Research Council of Finland, project number 369424.

Declaration on Generative AI

The authors have not employed any Generative AI tools.

References

[1] J. Łukasiewicz, A. Tarski, Untersuchungen über den Aussagenkalküls, Comptes rendus de la Société des Sciences et des Lettres de Varsovie 23 (1930) 1–21.
[2] P. Hájek, Fuzzy logic and arithmetical hierarchy, Fuzzy sets and Systems 73 (1995) 359–363.

[3] J. Pavelka, On Fuzzy Logic I. Many-valued rules of inference, Mathematical Logic Quarterly 25 (1979) 45–52. URL: https://doi.org/10.1002/malq.19790250304. doi:10.1002/MALQ.19790250304.
[4] J. Pavelka, On Fuzzy Logic II. Enriched residuated lattices and semantics of propositional calculi, Mathematical Logic Quarterly 25 (1979) 119–134. URL: https://doi.org/10.1002/malq.19790250706. doi:10.1002/MALQ.19790250706.
[5] J. Pavelka, On Fuzzy Logic III. Semantical completeness of some many-valued propositional calculi, Mathematical Logic Quarterly 25 (1979) 447–464. URL: https://doi.org/10.1002/malq.19790252510. doi:10.1002/MALQ.19790252510.
[6] P. Hájek, Metamathematics of Fuzzy Logic, volume 4 of *Trends in Logic*, Kluwer, 1998. URL: https://doi.org/10.1007/978-94-011-5300-3. doi:10.1007/978-94-011-5300-3.
[7] F. Esteva, L. Godo, F. Montagna, The LII and $LII\frac{1}{2}$ logics: two complete fuzzy systems joining Łukasiewicz and Product Logics, Archive for Mathematical Logic 40 (2001) 39–67.
[8] R. McNaughton, A Theorem About Infinite-Valued Sentential Logic, The Journal of Symbolic Logic 16 (1951) 1–13.
[9] P. Amato, A. Di Nola, B. Gerla, Neural Networks and Rational McNaughton Functions., Journal of Multiple-Valued Logic and Soft Computing 11 (2005) 95–110.
[10] A. Di Nola, B. Gerla, I. Leustean, Adding real coefficients to Łukasiewicz logic: An application to neural networks, in: International Workshop on Fuzzy Logic and Applications, Springer, 2013, pp. 77–85.
[11] J. L. Castro, E. Trillas, The Logic of Neural Networks, Mathware and Soft Computing 5 (1998) 23–37.
[12] W. S. McCulloch, W. Pitts, A logical calculus of the ideas immanent in nervous activity, The bulletin of mathematical biophysics 5 (1943) 115–133.
[13] V. Ahvonen, D. Heiman, A. Kuusisto, Descriptive Complexity for Neural Networks via Boolean Networks, in: A. Murano, A. Silva (Eds.), 32nd EACSL Annual Conference on Computer Science Logic, CSL 2024, Naples, Italy, February 19-23, 2024, volume 288 of *LIPIcs*, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2024, pp. 9:1–9:22. URL: https://doi.org/10.4230/LIPIcs.CSL.2024.9. doi:10.4230/LIPICS.CSL.2024.9.
[14] V. Ahvonen, D. Heiman, A. Kuusisto, Descriptive complexity for neural networks via Boolean networks, Journal of Logic and Computation 36 (2026) exag011. URL: https://doi.org/10.1093/logcom/exag011. doi:10.1093/logcom/exag011.
[15] E. Marchioni, F. Montagna, Complexity and Definability Issues in $\text{ŁII}\frac{1}{2}$, Journal of Logic and Computation 17 (2007) 311–331. URL: https://doi.org/10.1093/logcom/exl044. doi:10.1093/LOGCOM/EXL044.
[16] P. Barceló, E. V. Kostylev, M. Monet, J. Pérez, J. Reutter, J. P. Silva, The Logical Expressiveness of Graph Neural Networks, in: International conference on learning representations, 2020.
[17] S. P. Hauke, P. A. Wałęga, Aggregate-Combine-Readout GNNs Can Express Logical Classifiers Beyond the Logic C2, Proceedings of the AAAI Conference on Artificial Intelligence 40 (2026) 21594–21601. URL: https://ojs.aaai.org/index.php/AAAI/article/view/39308. doi:10.1609/aaai.v40i26.39308.
[18] M. Grohe, The Descriptive Complexity of Graph Neural Networks, TheoretiCS 3 (2024).
[19] M. Benedikt, C.-H. Lu, B. Motik, T. Tan, Decidability of Graph Neural Networks via Logical Characterizations, in: K. Bringmann, M. Grohe, G. Puppis, O. Svensson (Eds.), 51st International Colloquium on Automata, Languages, and Programming (ICALP 2024), volume 297 of *Leibniz International Proceedings in Informatics (LIPIcs)*, Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl, Germany, 2024, pp. 127:1–127:20. URL: https://drops.dagstuhl.de/entities/document/10.4230/LIPIcs.ICALP.2024.127. doi:10.4230/LIPIcs.ICALP.2024.127.
[20] P. Nunn, M. Sälzer, F. Schwarzentruber, N. Troquard, A Logic for Reasoning about Aggregate-Combine Graph Neural Networks, in: K. Larson (Ed.), Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence, IJCAI-24, International Joint Conferences on Artificial Intelligence Organization, 2024, pp. 3532–3540. URL: https://doi.org/10.24963/ijcai.2024/391. doi:10.24963/ijcai.2024/391, main Track.

[21] B. C. Grau, E. Feng, P. A. Wałęga, The Correspondence Between Bounded Graph Neural Networks and Fragments of First-Order Logic, Proceedings of the AAAI Conference on Artificial Intelligence 40 (2026) 19135–19142. URL: https://ojs.aaai.org/index.php/AAAI/article/view/38987. doi:10.1609/aaai.v40i23.38987.
[22] M. Schönherr, C. Lutz, Logical Characterizations of GNNs with Mean Aggregation, Proceedings of the AAAI Conference on Artificial Intelligence 40 (2026) 25218–25225. URL: https://ojs.aaai.org/index.php/AAAI/article/view/39713. doi:10.1609/aaai.v40i30.39713.
[23] P. A. Wałęga, B. C. Grau, Preservation Theorems for Unravelling-Invariant Classes: A Uniform Approach for Modal Logics and Graph Neural Networks, arXiv preprint arXiv:2602.01856 (2026).
[24] H. Luo, J. Virtema, Unifying approach to uniform expressivity of graph neural networks, 2026. URL: https://arxiv.org/abs/2602.18409. arXiv:2602.18409.
[25] M. Pflueger, D. T. Cucala, E. V. Kostylev, Recurrent Graph Neural Networks and Their Connections to Bisimulation and Logic, in: Proceedings of the AAAI Conference on Artificial Intelligence, volume 38, 2024, pp. 14608–14616.
[26] V. Ahvonen, D. Heiman, A. Kuusisto, C. Lutz, Logical Characterizations of Recurrent Graph Neural Networks with Reals and Floats, Advances in Neural Information Processing Systems 37 (2024) 104205–104249.
[27] M. Selin, Descriptive Complexity and Graph Neural Networks, Master's thesis, Tampere University, 2025. URL: https://trepo.tuni.fi/handle/10024/232168.
[28] F. Esteva, L. Godo, C. Noguera, First-order t-norm based fuzzy logics with truth-constants: distinguished semantics and completeness properties, Annals of Pure and Applied Logic 161 (2009) 185–202.
[29] P. Savický, R. Cignoli, F. Esteva, L. Godo, C. Noguera, On Product Logic with Truth-constants, Journal of Logic and Computation 16 (2006) 205–225.
[30] F. Esteva, L. Godo, P. Hájek, M. Navara, Residuated fuzzy logics with an involutive negation, Archive for mathematical logic 39 (2000) 103–124.
[31] R. Horčík, P. Cintula, Product Łukasiewicz Logic, Archive for Mathematical Logic 43 (2004) 477–503.
[32] C. C. Chang, Algebraic Analysis of Many Valued Logics, Transactions of the American Mathematical society 88 (1958) 467–490.
[33] D. Mundici, Advanced Łukasiewicz calculus and MV-algebras, volume 35, Springer Science & Business Media, 2011.

A. Proof of Lemma 2.2

In this section, we prove Lemma 2.2:

Lemma 2.2. *For all terms t of $\mathbb{Q}[\mathrm{ReLU}, x_i]_{i\in\mathbb{N}}$, t is a proto-neuron if and only if $\deg(t) \leq 1$.*

Proof. We prove the claim by induction over the structure of t. If $t = \overline{r}$ for some $r \in \mathbb{Q}$ or if $t = x$ for some $x \in \mathrm{VAR}$, then t is a proto-neuron by definition. Next, assume that t' and t'' are terms of $\mathbb{Q}[\mathrm{ReLU}, x_i]_{i\in\mathbb{N}}$ and that t' and t'' are proto-neurons if and only if $\deg(t') \leq 1$ and $\deg(t'') \leq 1$, respectively.

- If $t = t' + t''$, then t is a proto-neuron if and only if t' and t'' are proto-neurons. By the induction hypothesis, this is equivalent to $\deg(t'), \deg(t'') \leq 1$ which in turn is equivalent to $\deg(t) = \max\{\deg(t'), \deg(t'')\} \leq 1$.
- If $t = t't''$, then t is a proto-neuron if and only if t' and t'' are proto-neurons such that we have $\deg(t') + \deg(t'') \leq 1$. Since $\deg(t) = \deg(t') + \deg(t'')$, this is equivalent to $\deg(t) \leq 1$.

- If $t = \mathrm{ReLU}(t')$, then t is a proto-neuron if and only if t' is a proto-neuron. By the induction hypothesis, this is equivalent to $\deg(t') \leq 1$, and since $\deg(t) = \deg(t')$, this is equivalent to $\deg(t) \leq 1$.

This concludes the induction. □

B. Defining rational numbers in $L\Pi\frac{1}{2}$

It is shown in [7] that all rational numbers in $[0,1]$ are definable in $L\Pi\frac{1}{2}$, meaning that for every $r \in \mathbb{Q} \cap [0,1]$ there exists a formula φ of $L\Pi\frac{1}{2}$ such that $V(\varphi) = r$ for each valuation V. We show an alternative construction to the one in [7], but the results of the current paper hold regardless of which way the construction is done. We define each rational number $r \in \mathbb{Q} \cap [0,1]$ in $L\Pi\frac{1}{2}$ as a formula $\underline{r}$ recursively as follows. First, we can obviously define $\underline{0} = \overline{0}$ and $\underline{\frac{1}{2}} = \overline{\frac{1}{2}}$. Second, we define $\underline{1} := \neg_L \overline{0}$. Next, we cover the case where $r = \frac{k}{2^j} \in]0,1[$ for some $k, j \in \mathbb{Z}_+$ such that $k < 2^j$. If $k = j = 1$, then $r = \frac{1}{2}$, and thus $\underline{r}$ is already defined. Next, assume that $j > 1$ and we have defined $\underline{\frac{k'}{2^{j-1}}}$ for each $k' \in \mathbb{Z}_+$ such that $k' < 2^{j-1}$. If $k = 1$, then we define

$$\underline{\frac{1}{2^j}} := \underline{\frac{1}{2^{j-1}}} \odot \underline{\frac{1}{2}}.$$

Next, assume that $k > 1$ and we have defined $\underline{\frac{k'}{2^j}}$ for each $k' \in \mathbb{Z}_+$ such that $k' < k$. If k is even, then $\frac{k}{2^j} = \frac{k'}{2^{j-1}}$ where $k' = \frac{k}{2} \in \mathbb{Z}_+$, meaning that $\underline{r}$ is already defined. If k is odd, then 2^j is not divisible by k, meaning that $\underline{r}$ is not yet defined. We define

$$\underline{\frac{k}{2^j}} := \underline{\frac{k-1}{2^j}} \oplus \underline{\frac{1}{2^j}}.$$

Lastly, we consider the case where $r = \frac{k}{\ell} \in]0,1[$ for some $k, \ell \in \mathbb{Z}_+$ such that $k < \ell$ and ℓ is not divisible by k nor is ℓ a power of 2. Let m be the smallest integer such that $k, \ell < 2^m$. Then

$$\underline{\frac{k}{\ell}} := \underline{\frac{\ell}{2^m}} \to_P \underline{\frac{k}{2^m}}.$$

This completes the construction.

Lemma B.1. *For each $r \in \mathbb{Q} \cap [0,1]$ and each valuation V, we have $V(\underline{r}) = r$.*

Proof. By straightforward induction over the recursive definition of $\underline{r}$ utilising the semantics of $\neg_L$, $\odot$, $\oplus$ and $\to_P$. □

C. Proof of Proposition 3.7

In this section, we prove Proposition 3.7. First we recall the proposition.

Proposition 3.7. *For all $n \in \mathbb{N}$, the logics $\mathrm{RPL}(\odot)_{\leq n}$ and $L\Pi\frac{1}{2}(\to_P^-)_{\leq n}$ have the same expressive power.*

We start with a lemma that will also prove useful in later sections.

Lemma C.1. *Let $\mathcal{L}$ be $\mathrm{RPL}(\odot)$ or $L\Pi\frac{1}{2}(\to_P^-)$. Let $i, j \in \mathbb{N}$, let φ be a formula of $\mathcal{L}_{\leq i}$ and let ψ be a formula of $\mathcal{L}_{\leq j}$. Then the following hold.*

- *For each* $r \in \mathbb{Q} \cap [0,1]$, $\overline{r}$ *is a formula of* $\mathrm{RPL}(\odot)_{\leq 0}$ *and* $\underline{r}$ *is a formula of* $L\Pi\frac{1}{2}(\to_P^-)_{\leq 0}$.
- *Each* $p \in \mathrm{PROP}$ *is a formula of* $\mathcal{L}_{\leq 1}$.
- $\neg_L \varphi$ *is a formula of* $\mathcal{L}_{\leq i}$.
- $\varphi \to_L \psi$, $\varphi \oplus \psi$, $\varphi \otimes \psi$ *and* $\varphi \ominus \psi$ *are formulae of* $\mathcal{L}_{\leq \max\{i,j\}}$.
- $\varphi \odot \psi$ *is a formula of* $\mathcal{L}_{\leq i+j}$.

Proof. The cases for $\overline{r}$, p, $\varphi \to_L \psi$ and $\varphi \odot \psi$ follow from the definition of the fragments $\mathcal{L}_{\leq n}$. The formula $\underline{r}$ is built without proposition symbols and thus a formula of $L\Pi\frac{1}{2}(\to_P^-)_{\leq 0}$. The cases for $\neg_L\varphi$, $\varphi \oplus \psi$, $\varphi \otimes \psi$ and $\varphi \ominus \psi$ are then easy to check by the definitions of the abbreviated connectives $\neg_L$, $\oplus$, $\otimes$ and $\ominus$, which are constructed using only the truth constant $\overline{0}$ and the implication $\to_L$. □

Now we are ready to prove the proposition.

Proposition 3.7. *For all* $n \in \mathbb{N}$, *the logics* $\mathrm{RPL}(\odot)_{\leq n}$ *and* $L\Pi\frac{1}{2}(\to_P^-)_{\leq n}$ *have the same expressive power.*

Proof. We show the claim by induction over n. The interesting part is the base case where $n = 0$, as the induction step is then trivial by the identical way in which the fragments $\mathrm{RPL}(\odot)_{\leq n}$ and $L\Pi\frac{1}{2}(\to_P^-)_{\leq n}$ are constructed from $\mathrm{RPL}(\odot)_{\leq 0}$ and $L\Pi\frac{1}{2}(\to_P^-)_{\leq 0}$, respectively. First, note that for each formula φ of either $\mathrm{RPL}(\odot)_{\leq 0}$ or $L\Pi\frac{1}{2}(\to_P^-)_{\leq 0}$, we have $V(\varphi) = V'(\varphi) = r \in \mathbb{Q} \cap [0,1]$ for all valuations V and V'; this is easy to see because φ contains no proposition symbols, which guarantees that the truth value of φ is a constant and indeed a rational number. Now, by Lemmas B.1 and C.1, if φ is a formula of $\mathrm{RPL}(\odot)_{\leq 0}$, then $\underline{r}$ is an equivalent formula of $L\Pi\frac{1}{2}(\to_P^-)_{\leq 0}$. Likewise, if φ is a formula of $L\Pi\frac{1}{2}(\to_P^-)_{\leq 0}$, then $\overline{r}$ is an equivalent formula of $\mathrm{RPL}(\odot)_{\leq 0}$. □

D. Auxiliary connectives

In this section, we establish some auxiliary connectives that will be useful when translating terms of $\mathbb{Q}[\mathrm{ReLU}, x_i]_{i\in\mathbb{N}}$ to the logics $\mathrm{RPL}(\odot)$ and $L\Pi\frac{1}{2}$. The goal with these auxiliary connectives is to simulate real arithmetic from some arbitrary interval $[-k, k]$ in the interval $[0,1]$. While we already have the connectives $\oplus$ and $\odot$ whose semantics are similar to the sum and multiplication of real numbers, this does not yet suffice for our characterisations. Consider for example the multiplication of two real numbers $a, b > 1$. We have $ab > a$ and $ab > b$. On the other hand, multiplying any two numbers $a, b \in [0,1]$, we get $ab \leq a$ and $ab \leq b$. Thus, $\odot$ by itself does not suffice to simulate multiplication of two real numbers, but we will see by the end of the section that a more involved auxiliary connective will accomplish the task.

To begin with, we first define for each $n \in \mathbb{N}$ the iterated sum operator $\bigodot_n$. Intuitively, $\bigodot_n$ corresponds to multiplying the truth value of a formula by n, which is achieved by iteratively summing the formula with itself. For $n = 0$ we define $\bigodot_0 \varphi := \overline{0}$. Then, for each $n \in \mathbb{N}$, we define

$$\bigodot_{n+1} \varphi := (\bigodot_n \varphi) \oplus \varphi.$$

Lemma D.1. *For all* $n \in \mathbb{N}$, *all valuations* V *and all formulae* φ *of* $\mathrm{RPL}(\odot)$ *or* $L\Pi\frac{1}{2}$, *we have that* $V(\bigodot_n \varphi) = \min\{1, nV(\varphi)\}$.

Proof. By straightforward induction over n. The details are left to the reader. □

For $\mathrm{RPL}(\odot)$ and its fragments, we introduce the connective $\overline{\oplus}'$. The intuition behind $\overline{\oplus}'$ is to simulate the addition of real numbers belonging to the interval $[-k, k]$ for any $k \in \mathbb{R}_+$ in the interval $[0, 1]$ using the scaling function scale_k. More formally, we define $\overline{\oplus}'$ as follows:

$$\varphi \mathbin{\overline{\oplus}'} \psi := \left(\varphi \ominus \overline{\tfrac{1}{4}}\right) \oplus \left(\psi \ominus \overline{\tfrac{1}{4}}\right).$$

For $L\Pi\frac{1}{2}$ and its fragments, where the truth constant $\overline{\frac{1}{4}}$ does not exist, we instead define the connective $\underline{\oplus}'$ which is defined the same as $\overline{\oplus}'$ except that we replace $\overline{\frac{1}{4}}$ with the formula $\underline{\frac{1}{4}}$ (recall Appendix B). In the rest of the paper, the logic is always clear from the context, and thus we just write $\oplus'$ instead of $\overline{\oplus}'$ or $\underline{\oplus}'$.

The following lemma shows that $\oplus'$ indeed simulates the addition of two real numbers in the interval $[-k, k]$ correctly, assuming that the inputs lie in the interval $\left[-\frac{k}{2}, \frac{k}{2}\right]$. The restriction of inputs is natural, since $a, b \in \left[-\frac{k}{2}, \frac{k}{2}\right]$ implies $a + b \in [-k, k]$.

Lemma D.2. *Let $\mathcal{L}$ be $\mathrm{RPL}(\odot)$ or $L\Pi\frac{1}{2}$. For all valuations V, all $k \in \mathbb{R}_+$ and all formulae φ and ψ of $\mathcal{L}$: if $V(\varphi) = \mathrm{scale}_k(\ell)$ and $V(\psi) = \mathrm{scale}_k(\ell')$ for some $\ell, \ell' \in \left[-\frac{k}{2}, \frac{k}{2}\right]$, then $V(\varphi \oplus' \psi) = \mathrm{scale}_k(\ell + \ell')$.*

Proof. We cover the case where $\mathcal{L}$ is $\mathrm{RPL}(\odot)$ as the case where $\mathcal{L}$ is $L\Pi\frac{1}{2}$ is analogous. By the semantics of $\ominus$, we have $V\left(\varphi \ominus \overline{\frac{1}{4}}\right) = \max\left\{0, V(\varphi) - V\left(\overline{\frac{1}{4}}\right)\right\}$. Because $V\left(\overline{\frac{1}{4}}\right) = \frac{1}{4}$ and

$$V(\varphi) = \mathrm{scale}_k(\ell) \geq \mathrm{scale}_k(-\tfrac{k}{2}) = \tfrac{k - \frac{k}{2}}{2k} = \tfrac{1}{4},$$

we have $V(\varphi) - V\left(\overline{\frac{1}{4}}\right) \geq \frac{1}{4} - \frac{1}{4} = 0$. Thus, we have $V\left(\varphi \ominus \overline{\frac{1}{4}}\right) = V(\varphi) - \frac{1}{4}$. Analogously, we see that $V\left(\psi \ominus \overline{\frac{1}{4}}\right) = V(\psi) - \frac{1}{4}$.

Now, by the semantics of $\oplus$ we have $V(\varphi \oplus' \psi) = \min\left\{1, V\left(\varphi \ominus \overline{\frac{1}{4}}\right) + V\left(\psi \ominus \overline{\frac{1}{4}}\right)\right\}$. By the above, we have $V\left(\varphi \ominus \overline{\frac{1}{4}}\right) + V\left(\psi \ominus \overline{\frac{1}{4}}\right) = \left(V(\varphi) - \frac{1}{4}\right) + \left(V(\psi) - \frac{1}{4}\right) = V(\varphi) + V(\psi) - \frac{1}{2}$, which implies $V(\varphi \oplus' \psi) = \min\left\{1, V(\varphi) + V(\psi) - \frac{1}{2}\right\}$. Because $\ell, \ell' \leq \frac{k}{2}$, we have

$$V(\varphi), V(\psi) \leq \mathrm{scale}_k\left(\tfrac{k}{2}\right) = \tfrac{k + \frac{k}{2}}{2k} = \tfrac{3}{4},$$

which means that

$$V(\varphi) + V(\psi) - \tfrac{1}{2} \leq \tfrac{3}{4} + \tfrac{3}{4} - \tfrac{1}{2} = 1.$$

Thus

$$\begin{aligned} V(\varphi \oplus' \psi) &= V(\varphi) + V(\psi) - \tfrac{1}{2} \\ &= \mathrm{scale}_k(\ell) + \mathrm{scale}_k(\ell') - \tfrac{1}{2} \\ &= \tfrac{k+\ell}{2k} + \tfrac{k+\ell'}{2k} - \tfrac{k}{2k} \\ &= \tfrac{k+(\ell+\ell')}{2k} \\ &= \mathrm{scale}_k(\ell + \ell'). \end{aligned}$$

This completes the proof. □

Remark D.3. The definition of $\oplus'$ may raise the question: why not simply define $\varphi \oplus' \psi := (\varphi \oplus \psi) \ominus \overline{\frac{1}{2}}$? The problem is that, while the formula would still give correct truth values when $V(\varphi) = \mathrm{scale}_k(\ell)$ and $V(\psi) = \mathrm{scale}_k(\ell')$ for some $\ell, \ell' \in \left[-\frac{k}{2}, 0\right]$, it would not give correct values when $\ell, \ell' \in \left]0, \frac{k}{2}\right]$, because in all such cases we would get $V(\varphi \oplus' \psi) = \frac{1}{2} = \mathrm{scale}_k(0) \neq \mathrm{scale}_k(\ell + \ell')$. The chosen definition of $\oplus'$ is intended to maximize the working range of the connective while also treating the two connected formulae φ and ψ symmetrically.

Analogously to the connective $\bigodot_n$, we define a second iterated sum operator $\bigodot'_n$ for each $n \in \mathbb{N}$, this time corresponding to the connective $\oplus'$. Again, the operator $\bigodot'_n$ corresponds to multiplying by an integer n, but this time the multiplication occurs in the larger interval $[-k, k]$ and the result is scaled down to $[0, 1]$. For $n = 0$, we define $\bigodot'_0 \varphi := \overline{\frac{1}{2}}$, and for $n = 1$, we define $\bigodot'_1 \varphi := \varphi$. Next, let $n > 1$ and assume we have defined $\bigodot'_m \varphi$ for each non-negative integer $m < n$. We define

$$\bigodot'_n \varphi := \left(\bigodot'_{\lceil \frac{n}{2} \rceil} \varphi\right) \oplus' \left(\bigodot'_{\lfloor \frac{n}{2} \rfloor} \varphi\right).$$

The next lemma shows that $\bigodot'_n$ correctly simulates the multiplication of a real number in the interval $[-k, k]$ by n, assuming that the input is in the interval $\left[-\frac{k}{n+1}, \frac{k}{n+1}\right]$. This restriction is natural, since $a \in \left[-\frac{k}{n+1}, \frac{k}{n+1}\right]$ implies $na \in \left[-\frac{nk}{n+1}, \frac{nk}{n+1}\right] \subset [-k, k]$. Note that simply restricting to $\left[-\frac{k}{n}, \frac{k}{n}\right]$ does not suffice because the left half of $\bigodot'_n \varphi$ accounts for more than half the value of the scaled sum when n is odd.

Lemma D.4. *Let $\mathcal{L}$ be $\mathrm{RPL}(\odot)$ or $L\Pi\frac{1}{2}$. Let $n \in \mathbb{N}$, $k \in \mathbb{R}_+$, let V be a valuation and φ a formula of $\mathcal{L}$ such that $V(\varphi) = \mathrm{scale}_k(\ell)$ for some $\ell \in \left[-\frac{k}{n+1}, \frac{k}{n+1}\right]$. Then $V(\bigodot'_n \varphi) = \mathrm{scale}_k(n\ell)$.*

Proof. We prove the claim by induction over n. If $n = 0$, then $\bigodot'_0 \varphi = \overline{\frac{1}{2}}$ and therefore we have $V(\bigodot'_0 \varphi) = \frac{1}{2} = \mathrm{scale}_k(0\ell)$. If $n = 1$, then $\bigodot'_1 \varphi = \varphi$ and $V(\bigodot'_1 \varphi) = V(\varphi) = \mathrm{scale}_k(1\ell)$. Next, assume that $n > 1$ and the claim holds for each $n' < n$. Assume that $V(\varphi) = \mathrm{scale}_k(\ell)$ for some $\ell \in \left[-\frac{k}{n+1}, \frac{k}{n+1}\right]$. By the definition of $\bigodot'_n$, we have $\bigodot'_n \varphi = \left(\bigodot'_{\lceil \frac{n}{2} \rceil} \varphi\right) \oplus' \left(\bigodot'_{\lfloor \frac{n}{2} \rfloor} \varphi\right)$. Because $\ell \in \left[-\frac{k}{n+1}, \frac{k}{n+1}\right]$ implies $\ell \in \left[-\frac{k}{\lceil \frac{n}{2} \rceil + 1}, \frac{k}{\lceil \frac{n}{2} \rceil + 1}\right]$ and $\ell \in \left[-\frac{k}{\lfloor \frac{n}{2} \rfloor + 1}, \frac{k}{\lfloor \frac{n}{2} \rfloor + 1}\right]$, by the induction hypothesis we have $V\left(\bigodot'_{\lceil \frac{n}{2} \rceil} \varphi\right) = \mathrm{scale}_k\left(\lceil \frac{n}{2} \rceil \ell\right)$ and $V\left(\bigodot'_{\lfloor \frac{n}{2} \rfloor} \varphi\right) = \mathrm{scale}_k\left(\lfloor \frac{n}{2} \rfloor \ell\right)$.

We want to use Lemma D.2, for which we need to show that $\lceil \frac{n}{2} \rceil \ell, \lfloor \frac{n}{2} \rfloor \ell \in \left[-\frac{k}{2}, \frac{k}{2}\right]$. As $\lceil \frac{n}{2} \rceil, \lfloor \frac{n}{2} \rfloor \leq \frac{n+1}{2}$ and $\ell \in \left[-\frac{k}{n+1}, \frac{k}{n+1}\right]$, we have $\lceil \frac{n}{2} \rceil \ell, \lfloor \frac{n}{2} \rfloor \ell \in \left[\frac{n+1}{2}(-\frac{k}{n+1}), \frac{n+1}{2}\frac{k}{n+1}\right] = \left[-\frac{k}{2}, \frac{k}{2}\right]$. Now by Lemma D.2, we get $V(\bigodot_n \varphi) = \mathrm{scale}_k(\lceil \frac{n}{2} \rceil \ell + \lfloor \frac{n}{2} \rfloor \ell) = \mathrm{scale}_k(n\ell)$, as desired. □

Lastly, for $\mathrm{RPL}(\odot)$ and its fragments, we define the connective $\overline{\odot}^k$, which is tied to an integer parameter $k \in \mathbb{Z}_+$. The intuition behind $\overline{\odot}^k$ is to simulate the multiplication of two real numbers belonging to the interval $[-k, k]$ in the interval $[0, 1]$. Formally, $\overline{\odot}^k$ is defined as follows:

$$\varphi \,\overline{\odot}^k\, \psi := \bigodot_k \left(\left(\bigodot_2 (\varphi \odot \psi) \oplus \overline{\frac{1}{2k}}\right) \ominus (\varphi \oplus' \psi)\right).$$

Again, for $L\Pi\frac{1}{2}$ and its fragments, we instead define $\underline{\odot}^k$ whose definition is otherwise the same as $\overline{\odot}^k$ except that we again replace the truth constant $\overline{\frac{1}{2k}}$ with the formula $\underline{\frac{1}{2k}}$. As with $\oplus'$, for the rest of the paper the logic is always clear from the context, and so we simply write $\odot^k$.

The next lemma shows that $\odot^k$ indeed correctly simulates the multiplication of reals in $[-k, k]$, assuming the inputs are in the interval $\left[-\sqrt{k}, \sqrt{k}\right]$. The restriction of inputs is once again natural, since $a, b \in \left[-\sqrt{k}, \sqrt{k}\right]$ implies $ab \in [-k, k]$.

Lemma D.5. *Let $\mathcal{L}$ be $\mathrm{RPL}(\odot)$ or $L\Pi\frac{1}{2}$. For all integers $k \geq 8$, valuations V and formulae φ and ψ of $\mathcal{L}$: if $V(\varphi) = \mathrm{scale}_k(\ell)$ and $V(\psi) = \mathrm{scale}_k(\ell')$ for some $\ell, \ell' \in \left[-\sqrt{k}, \sqrt{k}\right]$, then $V(\varphi \odot^k \psi) = \mathrm{scale}_k(\ell\ell')$.*

Proof. We show the case where $\mathcal{L}$ is $\mathrm{RPL}(\odot)$ as the case where $\mathcal{L}$ is $L\Pi\frac{1}{2}$ is analogous. For convenience, we let $\theta_{\text{left}} := \bigodot_2(\varphi \odot \psi) \oplus \overline{\frac{1}{2k}}$ and $\theta_{\text{right}} := \varphi \oplus' \psi$. This means that $\varphi \odot^k \psi = \bigodot_k(\theta_{\text{left}} \ominus \theta_{\text{right}})$. We start by calculating $V(\theta_{\text{left}})$, for which we must first calculate $V(\bigodot_2(\varphi \odot \psi))$ and $V(\varphi \odot \psi)$. First, by definition we get

$$V(\varphi \odot \psi) = V(\varphi)V(\psi) = \mathrm{scale}_k(\ell)\mathrm{scale}_k(\ell') = \tfrac{k+\ell}{2k}\tfrac{k+\ell'}{2k} = \tfrac{k^2+k(\ell+\ell')+\ell\ell'}{4k^2}.$$

By Lemma D.1, we have $V(\bigodot_2(\varphi \odot \psi)) = \min\{1, 2V(\varphi \odot \psi)\}$. Because $\ell, \ell' \in \left[-\sqrt{k}, \sqrt{k}\right]$ we find that

$$\begin{aligned} 2V(\varphi \odot \psi) &= 2\tfrac{k^2+k(\ell+\ell')+\ell\ell'}{4k^2} \\ &= \tfrac{k^2+k(\ell+\ell')+\ell\ell'}{2k^2} \\ &\leq \tfrac{k^2+k\left(\sqrt{k}+\sqrt{k}\right)+\sqrt{k}\sqrt{k}}{2k^2} \\ &= \tfrac{k^2+2k\sqrt{k}+k}{2k^2} \\ &= \tfrac{k+2\sqrt{k}+1}{2k}, \end{aligned}$$

which is at most 1 when $k \geq 3 + 2\sqrt{2} \approx 5.83$. Because of the assumption that $k \geq 8$, we have $2V(\varphi \odot \psi) \leq 1$ and thus $V(\bigodot_2(\varphi \odot \psi)) = \frac{k^2+k(\ell+\ell')+\ell\ell'}{2k^2}$. Next, by the semantics of $\oplus$ we have $V(\theta_{\text{left}}) = \min\left\{1, V(\bigodot_2(\varphi \odot \psi)) + V\left(\overline{\frac{1}{2k}}\right)\right\}$. Utilising the analysis above, we find that

$$\begin{aligned} V(\bigodot_2(\varphi \odot \psi)) + V\left(\overline{\tfrac{1}{2k}}\right) &= \tfrac{k^2+k(\ell+\ell')+\ell\ell'}{2k^2} + \tfrac{1}{2k} \\ &\leq \tfrac{k+2\sqrt{k}+1}{2k} + \tfrac{1}{2k} \\ &= \tfrac{k+2\sqrt{k}+2}{2k}, \end{aligned}$$

which is at most 1 when $k \geq 4 + 2\sqrt{3} \approx 7.46$. Again, because of the assumption $k \geq 8$, we see that $V(\bigodot_2(\varphi \odot \psi)) + V\left(\overline{\frac{1}{2k}}\right) \leq 1$ and thus

$$V(\theta_{\text{left}}) = \tfrac{k^2+k(\ell+\ell')+\ell\ell'}{2k^2} + \tfrac{1}{2k} = \tfrac{k^2+k(\ell+\ell')+k+\ell\ell'}{2k^2}.$$

Next, we calculate $V(\theta_{\text{right}})$. We see that $\sqrt{k} \leq \frac{k}{2}$ when $k \geq 4$ and thus $\ell, \ell' \in \left[-\sqrt{k}, \sqrt{k}\right]$ implies $\ell, \ell' \in \left[-\frac{k}{2}, \frac{k}{2}\right]$, which means we can apply Lemma D.2 to θ_{right}, giving us

$$V(\theta_{\text{right}}) = \mathrm{scale}_k(\ell + \ell') = \tfrac{k+(\ell+\ell')}{2k} = \tfrac{k^2+k(\ell+\ell')}{2k^2}.$$

We finish the proof by calculating $V(\varphi \odot^k \psi)$, for which we first calculate $V(\theta_{\text{left}} \ominus \theta_{\text{right}})$. By the semantics of $\ominus$, we have $V(\theta_{\text{left}} \ominus \theta_{\text{right}}) = \max\{0, V(\theta_{\text{left}}) - V(\theta_{\text{right}})\}$. We see that

$$V(\theta_{\text{left}}) - V(\theta_{\text{right}}) = \tfrac{k^2+k(\ell+\ell')+k+\ell\ell'}{2k^2} - \tfrac{k^2+k(\ell+\ell')}{2k^2} = \tfrac{k+\ell\ell'}{2k^2} \geq \tfrac{k-\sqrt{k}\sqrt{k}}{2k^2} = 0,$$

which means that $V(\theta_{\text{left}} \ominus \theta_{\text{right}}) = V(\theta_{\text{left}}) - V(\theta_{\text{right}}) = \frac{k+\ell\ell'}{2k^2}$. Lastly, by Lemma D.1, we have $V(\varphi \odot^k \psi) = \min\{1, k\,V(\theta_{\text{left}} \ominus \theta_{\text{right}})\} = \min\left\{1, \frac{k+\ell\ell'}{2k}\right\}$. Because $\ell, \ell' \in \left[-\sqrt{k}, \sqrt{k}\right]$, we get $\frac{k+\ell\ell'}{2k} \leq \frac{k+\sqrt{k}\sqrt{k}}{2k} = 1$, which gives us $V(\varphi \odot^k \psi) = \frac{k+\ell\ell'}{2k} = \mathrm{scale}_k(\ell\ell')$, as desired. □

The following lemma extends Lemma C.1 to the auxiliary connectives defined in this section, relating them to the fragments of $\mathrm{RPL}(\odot)$ and $L\Pi\frac{1}{2}(\to_P^-)$ defined in Section 2.2.5. Importantly, it shows that the scaled sum $\oplus'$ and multiplication $\odot^k$ do not differ from the ordinary sum $\oplus$ and multiplication $\odot$ in terms of how they relate to the fragments.

Lemma D.6. *Let $\mathcal{L}$ be $\mathrm{RPL}(\odot)$ or $L\Pi\frac{1}{2}(\to_P^-)$. Let $i, j \in \mathbb{N}$, and let φ be a formula of $\mathcal{L}_{\leq i}$ and ψ a formula of $\mathcal{L}_{\leq j}$. Then the following hold.*

- *For each $n \in \mathbb{N}$, $\bigodot_n \varphi$ and $\bigodot'_n \varphi$ are formulae of $\mathcal{L}_{\leq i}$.*
- *$\varphi \oplus' \psi$ is a formula of $\mathcal{L}_{\leq \max\{i,j\}}$.*
- *For each $k \in \mathbb{Z}_+$, $\varphi \odot^k \psi$ is a formula of $\mathcal{L}_{\leq i+j}$.*

Proof. The proof is straightforward by the definitions of the operators and the fragments $\mathcal{L}_{\leq n}$ and by Lemma C.1. □

E. Proof of Theorem 4.1

In this section, we prove Theorem 4.1. First, we recall the theorem.

Theorem 4.1. *$\mathbb{Q}[\mathrm{ReLU}, x_i]_{i\in\mathbb{N}}$ has the same expressive power (w.r.t. scaling) as $\mathrm{RPL}(\odot)$ and $L\Pi\frac{1}{2}(\to_P^-)$.*

We start by defining two auxiliary notions regarding terms of $\mathbb{Q}[\mathrm{ReLU}, x_i]_{i\in\mathbb{N}}$: atomic length and subterm. Intuitively, subterms are those components of a term that are themselves terms, and atomic length refers to how many of these subterms are atomic (i.e., either rational constants or variables). More formally, we define the **atomic length** of a term t, denoted by $\mathrm{length}(t)$, as follows:

- If $t = \overline{r}$ for some $r \in \mathbb{Q}$ or $t = x$ for some $x \in \mathrm{VAR}$, then $\mathrm{length}(t) = 1$.
- If $t = t' + t''$ or $t = t't''$ for terms t' and t'', then $\mathrm{length}(t) = \mathrm{length}(t') + \mathrm{length}(t'')$.
- If $t = \mathrm{ReLU}(t')$ for some term t', then $\mathrm{length}(t) = \mathrm{length}(t')$.

Likewise, we define the set $\mathrm{subt}(t)$ of **subterms** of a term t as follows:

- If $t = \overline{r}$ for some $r \in \mathbb{Q}$ or $t = x$ for some $x \in \mathrm{VAR}$, then $\mathrm{subt}(t) = \{t\}$.
- If $t = t' + t''$ or $t = t't''$ for terms t' and t'', then $\mathrm{subt}(t) = \mathrm{subt}(t') \cup \mathrm{subt}(t'') \cup \{t\}$.
- If $t = \mathrm{ReLU}(t')$ for some term t', then $\mathrm{subt}(t) = \mathrm{subt}(t') \cup \{t\}$.

Before proving the theorem, we establish two lemmas. In the first lemma, we give a constructive translation from terms of $\mathbb{Q}[\mathrm{ReLU}, x_i]_{i\in\mathbb{N}}$ to formulae of $\mathrm{RPL}(\odot)$ and $L\Pi\frac{1}{2}(\to_P^-)$.

Lemma E.1. *Let $\mathcal{L}$ be $\mathrm{RPL}(\odot)$ or $L\Pi\frac{1}{2}(\to_P^-)$. For each term t of $\mathbb{Q}[\mathrm{ReLU}, x_i]_{i\in\mathbb{N}}$ and each $i \in \mathbb{R}_+$, there exists some $K \in \mathbb{Z}_+$ such that for each integer $k \geq K$ we can construct an (i, k)-equivalent formula φ_t of $\mathcal{L}$. For all $d \in \mathbb{N}$, if $\deg(t) \leq d$, then φ_t is a formula of $\mathcal{L}_{\leq d}$.*

Proof. The strategy is to recursively build an (i, k)-equivalent formula for each subterm of t including t itself. We show the proof for the case where $\mathcal{L}$ is $\mathrm{RPL}(\odot)$. The case where $\mathcal{L}$ is $L\Pi\frac{1}{2}(\to_P^-)$ follows by Propositions 3.7 and 3.8.

Let t be a term and let $i \in \mathbb{R}_+$. Let $r_{\max}$ be the largest rational number such that $\overline{r_{\max}}$ or $\overline{-r_{\max}}$ is a subterm of t and let $j := \max\{i, r_{\max}, 2\}$. For subterms t' of t, it is easy to show by induction over the structure of t' that given any evaluation map E such that $E(x) \in [-i, i]$ for all $x \in \mathrm{VAR}$, we have $|E(t')| \leq j^{\mathrm{length}(t')}$. Now let $K := \max\{j^{2\mathrm{length}(t)}, 8\}$ and let $k \geq K$ be an integer.

For each subterm s of t, we recursively construct a formula φ_s that is (i, k)-equivalent to s as follows:

- If $s = \overline{r}$ for some $r \in \mathbb{Q}$, then $|r| \leq r_{\max} \leq j \leq K \leq k$, which implies $\mathrm{scale}_k(r) \in [0,1]$. Moreover, because $k, r \in \mathbb{Q}$, we must have $\mathrm{scale}_k(r) = \frac{k+r}{2k} \in \mathbb{Q}$ and thus $\mathrm{scale}_k(r) \in \mathbb{Q} \cap [0,1]$. We define $\varphi_s := \overline{\mathrm{scale}_k(r)}$.
- If $s = x$ for some $x \in \mathrm{VAR}$, then $\varphi_s := p_x$.
- If $s = t' + t''$ for some terms t' and t'', then $\varphi_s := \varphi_{t'} \oplus' \varphi_{t''}$.
- If $s = t't''$ for some terms t' and t'', then $\varphi_s := \varphi_{t'} \odot^k \varphi_{t''}$ (which is defined because $k \in \mathbb{Z}_+$).
- If $s = \mathrm{ReLU}(t')$ for some term t', then $\varphi_s := \left(\varphi_{t'} \ominus \overline{\frac{1}{2}}\right) \oplus \overline{\frac{1}{2}}$.

We now prove the correctness of our construction. Let s be a subterm of t, let E be an evaluation map such that $E(x) \in [-i, i]$ for each $x \in \mathrm{VAR}$, and let V be the valuation such that $V(p) = \mathrm{scale}_k(E(x_p))$ for each $p \in \mathrm{PROP}$. We have to show $V(\varphi_s) = \mathrm{scale}_k(E(s))$. We prove the claim by induction over the structure of s. First, if $s = \overline{r}$ for some $r \in \mathbb{Q}$, then $V(\varphi_s) = V\left(\overline{\mathrm{scale}_k(r)}\right) = \mathrm{scale}_k(r) = \mathrm{scale}_k(E(s))$. Next, if $s = x$ for some $x \in \mathrm{VAR}$, then the claim is true by the definition of the valuation V.

Next, assume that the claim holds for terms t' and t'', i.e., assume that $V(\varphi_{t'}) = \mathrm{scale}_k(E(t'))$ and $V(\varphi_{t''}) = \mathrm{scale}_k(E(t''))$. Before delving into further steps, we show that the conditions of Lemmas D.2 and D.5 are met. First, we already know that $|E(t')| \leq j^{\mathrm{length}(t')}$. Because $\mathrm{length}(t') \leq \mathrm{length}(t)$, we have $|E(t')| \leq j^{\mathrm{length}(t)}$. Since $j^{\mathrm{length}(t)} = \sqrt{j^{2\mathrm{length}(t)}}$, we get $|E(t')| \leq \sqrt{j^{2\mathrm{length}(t)}}$. Now, because $k \geq K \geq j^{2\mathrm{length}(t)}$, we have $|E(t')| \leq \sqrt{k}$. Analogously, we see that $|E(t'')| \leq \sqrt{k}$. This means that $E(t'), E(t'') \in \left[-\sqrt{k}, \sqrt{k}\right]$. Furthermore, since $k \geq K \geq 8 > 4$, we have $\sqrt{k} < \frac{k}{2} < k$, which means that $E(t'), E(t'') \in \left[-\frac{k}{2}, \frac{k}{2}\right]$ and $E(t'), E(t'') \in [-k, k]$. Now, we are ready to cover the remaining cases.

- If $s = t' + t''$, then $\varphi_s = \varphi_{t'} \oplus' \varphi_{t''}$. Because it is the case that $E(t'), E(t'') \in \left[-\frac{k}{2}, \frac{k}{2}\right]$, we may apply Lemma D.2, which gives us $V(\varphi_s) = \mathrm{scale}_k(E(t') + E(t''))$. Now because $E(t') + E(t'') = E(t' + t'') = E(s)$, we have $V(\varphi_s) = \mathrm{scale}_k(E(s))$.
- If $s = t't''$, then $\varphi_s = \varphi_{t'} \odot^k \varphi_{t''}$. Because it is the case that $E(t'), E(t'') \in \left[-\sqrt{k}, \sqrt{k}\right]$, by Lemma D.5 we get $V(\varphi_s) = \mathrm{scale}_k(E(t')E(t''))$. Because $E(t')E(t'') = E(t't'') = E(s)$, we have $V(\varphi_s) = \mathrm{scale}_k(E(s))$.
- If $s = \mathrm{ReLU}(t')$, then $\varphi_s = \left(\varphi_{t'} \ominus \overline{\frac{1}{2}}\right) \oplus \overline{\frac{1}{2}}$. By the semantics of $\oplus$, $\ominus$ and $\overline{\frac{1}{2}}$, we have

$$\begin{aligned} V(\varphi_s) &= V\left(\left(\varphi_{t'} \ominus \overline{\tfrac{1}{2}}\right) \oplus \overline{\tfrac{1}{2}}\right) \\ &= \min\left\{1, V\left(\varphi_{t'} \ominus \overline{\tfrac{1}{2}}\right) + \tfrac{1}{2}\right\} \\ &= \min\left\{1, \max\left\{0, V\left(\varphi_{t'}\right) - \tfrac{1}{2}\right\} + \tfrac{1}{2}\right\}. \end{aligned}$$

  Because $\max\{a, b\} + c = \max\{a + c, b + c\}$ for all $a, b, c \in \mathbb{R}$, we get

$$V(\varphi_s) = \min\left\{1, \max\left\{\tfrac{1}{2}, V\left(\varphi_{t'}\right)\right\}\right\}.$$

  By the induction hypothesis, we have $V\left(\varphi_{t'}\right) = \mathrm{scale}_k(E(t'))$ and thus

$$V(\varphi_s) = \min\left\{1, \max\left\{\tfrac{1}{2}, \mathrm{scale}_k(E(t'))\right\}\right\}.$$

  Because $E(t') \in [-k, k]$, we must have $\mathrm{scale}_k(E(t')) \in [0,1]$ which in turn implies that $\max\left\{\frac{1}{2}, \mathrm{scale}_k(E(t'))\right\} \leq 1$ and thus

$$V(\varphi_s) = \max\left\{\tfrac{1}{2}, \mathrm{scale}_k(E(t'))\right\}.$$

  There are two cases to explore: $E(t') < 0$ and $E(t') \geq 0$.

1. If $E(t') < 0$, then $\mathrm{scale}_k(E(t')) < \frac{1}{2}$ and therefore $V(\varphi_s) = \frac{1}{2}$. Because $\frac{1}{2} = \mathrm{scale}_k(0)$, we have $V(\varphi_s) = \mathrm{scale}_k(0)$. Because $E(t') < 0$, we have $E(s) = E(\mathrm{ReLU}(t')) = 0$, which gives us $V(\varphi_s) = \mathrm{scale}_k(E(s))$.
2. If $E(t') \geq 0$, then $\mathrm{scale}_k(E(t')) \geq \frac{1}{2}$ and thus $V(\varphi_s) = \mathrm{scale}_k(E(t'))$. Finally, because $E(t') \geq 0$, we have $E(s) = E(\mathrm{ReLU}(t')) = E(t')$ and therefore $V(\varphi_s) = \mathrm{scale}_k(E(s))$.

We have now shown the correctness of our construction. The claim "For all $d \in \mathbb{N}$, if $\deg(t) \leq d$, then φ_t is a formula of $\mathcal{L}_{\leq d}$" now follows from the definition of degree, our construction and Lemmas C.1 and D.6 by a simple induction over the structure of t. □

In the next lemma, we give an opposite translation from formulae of $\mathrm{RPL}(\odot)$ and $L\Pi\frac{1}{2}(\to_P^-)$ to terms of $\mathbb{Q}[\mathrm{ReLU}, x_i]_{i\in\mathbb{N}}$.

Lemma E.2. *Let $\mathcal{L}$ be $\mathrm{RPL}(\odot)$ or $L\Pi\frac{1}{2}(\to_P^-)$. For each formula φ of $\mathcal{L}$, we can construct an equivalent term t_φ of $\mathbb{Q}[\mathrm{ReLU}, x_i]_{i\in\mathbb{N}}$. For all $d \in \mathbb{N}$, if φ is a formula of $\mathcal{L}_{\leq d}$, then $\deg(t_\varphi) \leq d$.*

Proof. The strategy is to recursively build an equivalent term for each formula φ of $\mathcal{L}$. We cover the case where $\mathcal{L}$ is $\mathrm{RPL}(\odot)$. The case where $\mathcal{L}$ is $L\Pi\frac{1}{2}(\to_P^-)$ follows by Propositions 3.7 and 3.8.

Let φ be a formula of $\mathrm{RPL}(\odot)$. We will construct a term t_φ equivalent to φ by induction over the structure of φ as follows. First, we cover the base cases.

- If $\varphi = \overline{r}$ for some $r \in \mathbb{Q} \cap [0, 1]$, then $t_\varphi = \overline{r}$.
- If $\varphi = p$ for some $p \in \mathrm{PROP}$, then $t_\varphi = x_p$.

Next, assume ψ and θ are formulae of $\mathrm{RPL}(\odot)$, and let t_ψ and t_θ be the corresponding terms of $\mathbb{Q}[\mathrm{ReLU}, x_i]_{i\in\mathbb{N}}$.

- If $\varphi = \psi \to_L \theta$, then $t_\varphi = \overline{1} - \mathrm{ReLU}(t_\psi - t_\theta)$.
- If $\varphi = \psi \odot \theta$, then $t_\varphi = t_\psi t_\theta$.

Next, we prove the correctness of the given construction. Let V be a valuation and let E be an evaluation map such that $E(x) = V(p_x)$ for all $x \in \mathrm{VAR}$. We have to show that $E(t_\varphi) = V(\varphi)$; we show this by induction over the structure of φ. First, we cover the base cases.

- If $\varphi = \overline{r}$ for some $r \in \mathbb{Q} \cap [0, 1]$, then $t_\varphi = \overline{r}$ and $E(t_\varphi) = E(\overline{r}) = r = V(\overline{r}) = V(\varphi)$.
- If $\varphi = p$ for some $p \in \mathrm{PROP}$, then $t_\varphi = x_p$ and $E(t_\varphi) = E(x_p) = V(p) = V(\varphi)$.

Next, assume the claim holds for formulae ψ and θ of $\mathrm{RPL}(\odot)$, i.e., $E(t_\psi) = V(\psi)$ and $E(t_\theta) = V(\theta)$.

- If $\varphi = \psi \to_L \theta$, then $t_\varphi = \overline{1} - \mathrm{ReLU}(t_\psi - t_\theta)$. By the semantics of ReLU, we thus have $E(t_\varphi) = 1 - \max\{0, E(t_\psi) - E(t_\theta)\}$. By the induction hypothesis, we know it is the case that $E(t_\varphi) = 1 - \max\{0, V(\psi) - V(\theta)\}$. Because $-\max\{a, b\} = \min\{-a, -b\}$ for all $a, b \in \mathbb{R}$, we have $E(t_\varphi) = 1 + \min\{0, -V(\psi) + V(\theta)\}$. Because $a + \min\{b, c\} = \min\{a + b, a + c\}$ for all $a, b, c \in \mathbb{R}$, we get $E(t_\varphi) = \min\{1, 1 - V(\psi) + V(\theta)\}$. Thus by the semantics of $\to_L$, we have $E(t_\varphi) = V(\varphi)$.
- If $\varphi = \psi \odot \theta$, then $t_\varphi = t_\psi t_\theta$. Thus $E(t_\varphi) = E(t_\psi t_\theta) = E(t_\psi)E(t_\theta)$. By the induction hypothesis, we have $E(t_\varphi) = V(\psi)V(\theta)$, and by the semantics of $\odot$, we get $E(t_\varphi) = V(\varphi)$.

We have now shown the correctness of our construction. The claim "For all $d \in \mathbb{N}$, if φ is a formula of $\mathcal{L}_{\leq d}$, then $\deg(t_\varphi) \leq d$" follows from Lemma C.1, our construction and the definition of degree by a straightforward induction over the structure of φ. □

Now, we are ready to prove Theorem 4.1.

Theorem 4.1. $\mathbb{Q}[\mathrm{ReLU}, x_i]_{i\in\mathbb{N}}$ *has the same expressive power (w.r.t. scaling) as* $\mathrm{RPL}(\odot)$ *and* $L\Pi\frac{1}{2}(\to_P^-)$.

Proof. Let $\mathcal{L}$ be $\mathrm{RPL}(\odot)$ or $L\Pi\frac{1}{2}(\to_P^-)$. First, let $(t_1, \dots, t_n)$ be a sequence of terms of $\mathbb{Q}[\mathrm{ReLU}, x_i]_{i\in\mathbb{N}}$. For each $i \in \mathbb{R}_+$, we need to find a sequence $(\varphi_1, \dots, \varphi_n)$ of formulae of $\mathcal{L}$ and some $k \in \mathbb{R}_+$ such that φ_j is (i, k)-equivalent to t_j for each $j \in \{1, \dots, n\}$. By Lemma E.1, for each $j \in \{1, \dots, n\}$ and for each $i \in \mathbb{R}_+$ there exists some $K_{i,j} \in \mathbb{Z}_+$ such that for each integer $k \geq K_{i,j}$ we can construct a formula that is (i, k)-equivalent to t_j. Let $K_i = \max\{K_{i,1}, \dots, K_{i,n}\}$. Now for each integer $k \geq K_i$ there exists for each $j \in \{1, \dots, n\}$ a formula φ_j that is (i, k)-equivalent to t_j, and $(\varphi_1, \dots, \varphi_n)$ is the tuple we seek.

Next, let $(\varphi_1, \dots, \varphi_n)$ be a sequence of formulae of $\mathcal{L}$. We have to find a sequence $(t_1, \dots, t_n)$ of terms such that for each $j \in \{1, \dots, n\}$, the term t_j and the formula φ_j are equivalent. To obtain this sequence, we simply apply Lemma E.2 to each formula φ_j to obtain the equivalent term t_j. □

F. Proof of Theorem 4.5

In this section, we give the proof of Theorem 4.5. We start by recalling the theorem.

Theorem 4.5. *Neural networks have the same expressive power (w.r.t. scaling) as the logics* $\mathrm{RPL}(\odot)_{\leq 1}$, $L\Pi\frac{1}{2}(\to_P^-)_{\leq 1}$, RPL *and* $L\Pi\frac{1}{2}(\odot^-, \to_P^-)$.

We first establish some auxiliary definitions. We say that two terms t and t' are $[0, 1]$-**equivalent** if $E(t) = E(t')$ for each evaluation map E such that $E(x) \in [0, 1]$ for each $x \in \mathrm{VAR}$. For two tuples of terms $T = (t_1, \dots, t_k)$ and $T' = (t'_1, \dots, t'_k)$ of equal length, we say that T and T' are $[0, 1]$-**equivalent** if t_i and t'_i are $[0, 1]$-equivalent for each $i \in \{1, \dots, k\}$.

We then establish two auxiliary lemmas concerning neural networks that will be needed in the translations. The first lemma states that we can arbitrarily increase the depth of a neural network.

Lemma F.1. *For each $d \in \mathbb{N}$, each neural network N of depth d and each $d' > d$, there exists a $[0, 1]$-equivalent neural network N' of depth d'. If $d \neq 0$, then N and N' are equivalent.*

Proof. Let $N = (n_1, \dots, n_k)$ be a neural network of depth d. We define $N' := (n'_1, \dots, n'_k)$ where $n'_j := \mathrm{ReLU}(\sum_{i=1}^k \overline{w_{i,j}} n_i + \overline{0})$ where $w_{j,j} = 1$ and $w_{i,j} = 0$ for each $i \neq j$. Now n'_j is equivalent to $\mathrm{ReLU}(n_j)$ and thus $[0, 1]$-equivalent to n_j for each $j \in \{1, \dots, k\}$. This is because each neuron n_j is either a variable (if $d = 0$) or of the form $\mathrm{ReLU}(t)$ for some proto-neuron t (if $d \neq 0$). In either case, for each evaluation map E such that $E(x) \in [0, 1]$ for each $x \in \mathrm{VAR}$, we have $E(n_j) \geq 0$ and thus $E(n'_j) = E(\mathrm{ReLU}(n_j)) = E(n_j)$. Thus, N' is a $[0, 1]$-equivalent neural network of depth $d + 1$ and we may iterate this process until we reach depth d'.

If $d \neq 0$, then n_j is of the form $\mathrm{ReLU}(t)$, and therefore for each evaluation map E we must have $E(n'_j) = E(\mathrm{ReLU}(n_j)) = E(n_j)$. Thus, n_j and n'_j are equivalent, which means that N and N' are equivalent. □

From the definition of neural networks, we see that a neural network of depth d is built on top of a neural network of depth $d - 1$ by adding additional neurons; thus, each neural network of depth d has embedded within it a neural network of each depth $0, \dots, d$. We say that two neural networks of depth at least d are **identical up to depth** d if their embedded neural networks of depth d are identical.

We next prove another auxiliary lemma, which intuitively shows that any number of neural networks can be modified such that the neural networks are pairwise identical apart from the output neurons.

Lemma F.2. *Given $k \in \mathbb{Z}_+$ and neural networks $N_1, \ldots, N_k$, there exist neural networks $N_1', \ldots, N_k'$ such that N_i and N_i' are $[0,1]$-equivalent for each $i \in \{1, \ldots, k\}$, and $N_1', \ldots, N_k'$ all have the same depth d and are identical up to depth $d-1$. For each $i \in \{1, \ldots, k\}$, if N_i has depth at least 1, then N_i and N_i' are equivalent.*

Proof. Intuitively, given $t \in \{1, \ldots, k\}$, we add every neuron from $N_1, \ldots, N_k$ to N_t', except for output neurons which are only added from N_t. We copy the weights and biases from $N_1, \ldots, N_k$, and we also add the weight $\overline{0}$ between neurons that are not from the same neural network, since our neural networks need to be fully connected. See Figure 4 for an illustrated example.

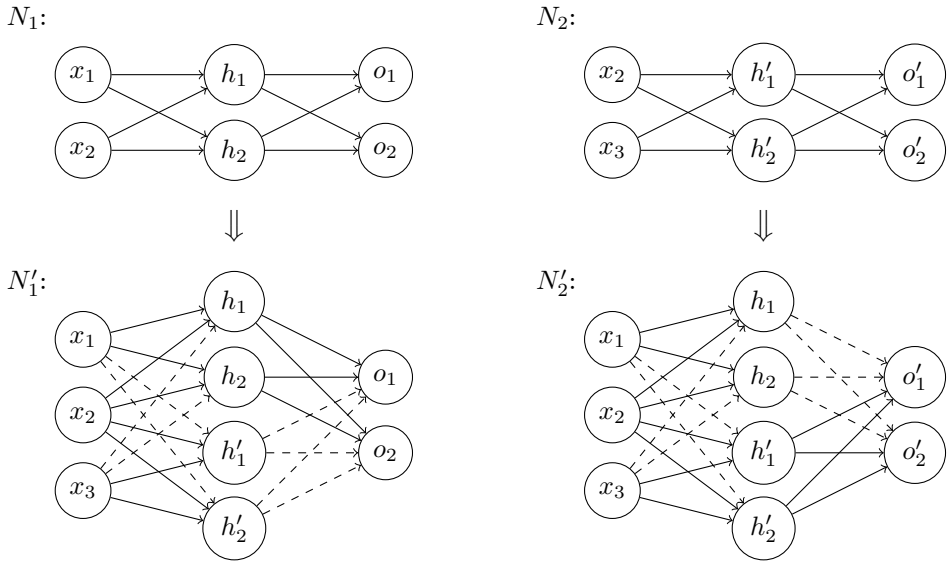

Figure 4: Top: Two neural networks N_1 and N_2. Bottom: Equivalent versions N_1' and N_2' where only the output neurons are different. Dashed lines indicate the weight 0 between two neurons.

Next, we give the construction formally. First, let d be the maximum depth of $N_1, \ldots, N_k$. By Lemma F.1, we can construct neural networks $M_1, \ldots, M_k$ of depth d such that M_j is $[0,1]$-equivalent to N_j for each $j \in \{1, \ldots, k\}$. Next, we construct the neural network N_t' that is equivalent to M_t, starting with the embedded neural network N_t^0 of depth 0 of N_t' and recursively moving to embedded neural networks $N_t^{d'}$ of increasing depths d' until we reach $N_t^d = N_t'$.

- First, we define $N_t^0 := (y_1, \ldots, y_{\ell_0})$, where $y_1, \ldots, y_{\ell_0}$ are the variables of the input layers of $M_1, \ldots, M_k$ with no duplicates (unless $d = 0$, in which case $y_1, \ldots, y_{\ell_0}$ are the variables of just M_t).
- Second, we define N_t^1. Let $(m_1, \ldots, m_\ell)$ be the concatenation of the embedded neural networks of depth 1 of $M_1, \ldots, M_k$ (unless $d = 1$ in which case let $(m_1, \ldots, m_\ell)$ be M_t). We define $N_t^1 = (n_1, \ldots, n_\ell)$, where $n_j = \mathrm{ReLU}\left(\sum_{i=1}^{\ell_0} \overline{w_{i,j}} y_i + \overline{b_j}\right)$ where $\overline{b_j}$ is the bias of m_j and $\overline{w_{i,j}}$ is the weight from y_i to m_j if they are from the same neural network and otherwise $\overline{w_{i,j}} = \overline{0}$.
- Next, we cover the case where $d' > 1$. Let $(m_1, \ldots, m_\ell)$ be the concatenation of the embedded neural networks of depth d' of $M_1, \ldots, M_k$. Let $N_t^{d'} = (n_1, \ldots, n_\ell)$ such that each n_j corresponds to the neuron m_j. Let $(m_1', \ldots, m_{\ell'}')$ be the concatenation of the embedded neural networks of depth $d'+1$ of $M_1, \ldots, M_k$ (unless, again, $d = d'+1$ in which case let $(m_1', \ldots, m_{\ell'}')$ be M_t). We define $N_t^{d'+1} := (n_1', \ldots, n_{\ell'}')$ where $n_j' = \mathrm{ReLU}\left(\sum_{i=1}^{\ell} \overline{w_{i,j}} n_i + \overline{b_j}\right)$ where $\overline{b_j}$ is the bias of

m'_j and $\overline{w_{i,j}}$ is the weight from m_i to m'_j if they are from the same neural network and otherwise $\overline{w_{i,j}} = \overline{0}$.

Lastly, we obviously define $N'_t := N_t^d$. The equivalence of M_t and N'_t is now clear to see, and because N_t and M_t are $[0,1]$-equivalent, N_t and N'_t are thus also $[0,1]$-equivalent. If N_t has depth at least 1, then N_t and M_t are equivalent, and thus N_t and N'_t are equivalent. With the exception of the output neurons, the construction was identical for each t, which means that $N'_1, \ldots, N'_k$ are pairwise identical up to depth $d-1$. □

Now, we are ready to show the translation from the logics $\mathrm{RPL}(\odot)_{\leq 1}$ and $L\Pi\frac{1}{2}(\to_P^-)_{\leq 1}$ to neurons.

Lemma F.3. *Let $\mathcal{L}$ be $\mathrm{RPL}(\odot)$ or $L\Pi\frac{1}{2}(\to_P^-)$. For each formula φ of $\mathcal{L}_{\leq 1}$, we can construct an equivalent neuron n_φ.*

Proof. We modify the proof of Lemma E.2 such that the constructed terms are, in fact, neurons. We cover the case where $\mathcal{L}$ is $\mathrm{RPL}(\odot)$. The case where $\mathcal{L}$ is $L\Pi\frac{1}{2}(\to_P^-)$ follows by Propositions 3.7 and 3.8.

Let φ be a formula of $\mathrm{RPL}(\odot)_{\leq 1}$. We will construct a neuron n_φ equivalent to φ by recursion over the structure of φ as follows. First, we cover the base cases.

- If $\varphi = p$ for some $p \in \mathrm{PROP}$, then $n_\varphi := x_p$.
- If φ is a formula of $\mathrm{RPL}(\odot)_{\leq 0}$, then there exists some $r \in \mathbb{Q} \cap [0,1]$ such that φ is equivalent to $\overline{r}$. We define $n_\varphi := \mathrm{ReLU}\left(\overline{0} \cdot x + \overline{r}\right)$.

Next, assume that ψ is a formula of $\mathrm{RPL}(\odot)_{\leq i}$ and θ a formula of $\mathrm{RPL}(\odot)_{\leq j}$ for some $i, j \in \{0,1\}$, and let n_ψ and n_θ be the equivalent neurons. Both n_ψ and n_θ can be seen as neural networks with a single output neuron, so by Lemma F.2 we can construct neurons n'_ψ and n'_θ that are $[0,1]$-equivalent to n_ψ and n_θ (and thus equivalent to ψ and θ) such that n'_ψ and n'_θ have the same depth d and are identical up to depth $d-1$.

- If $i + j \leq 1$ and $\varphi = \psi \odot \theta$, then without loss of generality we may assume that $i = 0$. Because ψ is now a formula of $\mathrm{RPL}(\odot)_{\leq 0}$, there exists some $r \in \mathbb{Q} \cap [0,1]$ such that ψ is equivalent to $\overline{r}$. We define $n_\varphi := \mathrm{ReLU}(\overline{r} \cdot n_\theta + \overline{0})$.
- If $\varphi = \psi \to_L \theta$, then we first define $m_\varphi := \mathrm{ReLU}((\overline{1} \cdot n'_\psi + \overline{-1} \cdot n'_\theta) + \overline{0})$, after which we define $n_\varphi := \mathrm{ReLU}(\overline{-1} \cdot m_\varphi + \overline{1})$.

Next, we prove the correctness of our construction. Let V be a valuation, and let E be the evaluation map such that $E(x) = V(p_x)$ for each $x \in \mathrm{VAR}$. We have to show that $E(n_\varphi) = V(\varphi)$.

- If $\varphi = p$ for some $p \in \mathrm{PROP}$, then $n_\varphi = x_p$ and the claim is trivially true.
- If φ is a formula of $\mathrm{RPL}(\odot)_{\leq 0}$, then $n_\varphi = \mathrm{ReLU}\left(\overline{0} \cdot x + \overline{r}\right)$ where $r \in \mathbb{Q} \cap [0,1]$ such that φ is equivalent to $\overline{r}$. Now $E(n_\varphi) = \max\{0, 0 \cdot E(x) + r\} = \max\{0, r\}$. Now $r \in \mathbb{Q} \cap [0,1]$ implies $r \geq 0$ and thus $E(n_\varphi) = r = V(\overline{r})$. Because φ is equivalent to $\overline{r}$, we have $V(\overline{r}) = V(\varphi)$ and thus $E(n_\varphi) = V(\varphi)$.

Next, assume that the claim holds for formulae ψ of $\mathrm{RPL}(\odot)_{\leq i}$ and θ of $\mathrm{RPL}(\odot)_{\leq j}$ where $i, j \in \{0,1\}$, let n_ψ and n_θ be the equivalent neurons, and let n'_ψ and n'_θ be as above.

- If $i + j \leq 1$ and $\varphi = \psi \odot \theta$, then without loss of generality we may assume that $i = 0$. Then $n_\varphi = \mathrm{ReLU}(\overline{r} \cdot n_\theta + \overline{0})$ where $r \in \mathbb{Q} \cap [0,1]$ such that ψ is equivalent to $\overline{r}$. Therefore we have $E(n_\varphi) = \max\{0, rE(n_\theta) + 0\} = \max\{0, rE(n_\theta)\}$. Because ψ is equivalent to $\overline{r}$, we have $r = V(\psi)$, and by the induction hypothesis we have $E(n_\theta) = V(\theta)$, and thus we

get $E(n_\varphi) = \max\{0, V(\psi)V(\theta)\} = \max\{0, V(\varphi)\}$. Lastly, because $V(\varphi) \in [0,1]$, we have $E(n_\varphi) = V(\varphi)$.

- If $\varphi = \psi \to_L \theta$, then $n_\varphi = \mathrm{ReLU}(\overline{-1} \cdot m_\varphi + \overline{1})$ where $m_\varphi = \mathrm{ReLU}((\overline{1} \cdot n'_\psi + \overline{-1} \cdot n'_\theta) + \overline{0})$. Therefore $E(n_\varphi) = \max\{0, -E(m_\varphi) + 1\}$ and $E(m_\varphi) = \max\{0, E(n'_\psi) - E(n'_\theta)\}$. By the induction hypothesis, $E(n'_\psi) = E(n_\psi) = V(\psi)$ and $E(n'_\theta) = E(n_\theta) = V(\theta)$ and therefore $E(m_\varphi) = \max\{0, V(\psi) - V(\theta)\}$. Because of the fact that $V(\psi), V(\theta) \in [0,1]$ we must have $E(m_\varphi) \leq \max\{0, 1-0\} = 1$ which implies $-E(m_\varphi) + 1 \geq -1 + 1 = 0$ and therefore $E(n_\varphi) = -E(m_\varphi) + 1$. Because $-\max\{a,b\} = \min\{-a,-b\}$ for all $a, b \in \mathbb{R}$, we have $-E(m_\varphi) = \min\{0, -V(\psi) + V(\theta)\}$ and thus $E(n_\varphi) = \min\{0, -V(\psi) + V(\theta)\} + 1$. Because $\min\{a,b\} + c = \min\{a+c, b+c\}$ for all $a, b, c \in \mathbb{R}$ and by the semantics of $\to_L$, we finally get $E(n_\varphi) = \min\{1, 1 - V(\psi) + V(\theta)\} = V(\psi \to_L \theta) = V(\varphi)$.

This completes the proof of correctness. □

Before delving into the other direction of the characterisation, we first establish a useful lemma.

Lemma F.4. *Let $k \in \mathbb{R}_+$, let φ be a formula of $\mathrm{RPL}(\odot)$ or $L\Pi\frac{1}{2}$, let V be a valuation and let $\ell \in [-k, k]$ such that $V(\varphi) = \mathrm{scale}_k(\ell)$. Then $V(\neg_L \varphi) = \mathrm{scale}_k(-\ell)$.*

Proof. By the semantics of $\neg_L$, $V(\neg_L \varphi) = 1 - V(\varphi) = 1 - \mathrm{scale}_k(\ell)$. By the definition of scale_k,

$$1 - \mathrm{scale}_k(\ell) = 1 - \tfrac{k+\ell}{2k} = \tfrac{2k}{2k} - \tfrac{k+\ell}{2k} = \tfrac{k-\ell}{2k} = \mathrm{scale}_k(-\ell),$$

and thus $V(\neg_L \varphi) = \mathrm{scale}_k(-\ell)$. □

Next we will show the other direction of the characterisation, where we translate proto-neurons into the logics RPL and $L\Pi\frac{1}{2}(\odot^-, \to_P^-)$. We split the translation into two parts presented as lemmas. In the first lemma, we replace the rationals of a proto-neuron with integers (although the lemma holds for not only proto-neurons but for all terms). In the second lemma, we translate the modified proto-neuron into a formula of RPL or $L\Pi\frac{1}{2}(\odot^-, \to_P^-)$.

Lemma F.5. *For each term t there exists some $D \in \mathbb{Z}_+$ such that we can construct a term s that satisfies the following:*

- *s is equivalent to $\overline{D}t$,*
- *s contains only integers, i.e., if s contains a subterm $\overline{r}$ for some $r \in \mathbb{Q}$, then $r \in \mathbb{Z}$, and*
- $\deg(s) = \deg(t)$.

Proof. To obtain s, we intuitively multiply t with an appropriate integer constant $\overline{D}$ (which can be constructively chosen based on the rational constants in t) and recursively cancel out the denominators of all the rationals in t. For example, if $t = \overline{\frac{5}{7}}\left(x + \overline{\frac{3}{7}}\right)$, then we choose $D = 7 \cdot 7 = 49$ and starting from $\overline{49}\,t = \overline{49}\left(\overline{\frac{5}{7}}\left(x + \overline{\frac{3}{7}}\right)\right)$, we obtain the following sequence of terms that are all equivalent to $\overline{49}\,t$: $s_0 = \overline{5}\left(\overline{7}\left(x + \overline{\frac{3}{7}}\right)\right)$, $s_1 = \overline{5}\left(\overline{7}\,x + \overline{7} \cdot \overline{\frac{3}{7}}\right)$ and $s_2 = \overline{5}\left(\overline{7}\,x + \overline{3}\right)$.

We now present the general case formally. We start with the observation that if $r \in \mathbb{Q}$, then $r = \frac{n_r}{d_r}$ for some numerator $n_r \in \mathbb{Z}$ and denominator $d_r \in \mathbb{Z}_+$ such that d_r is not divisible by n_r.

Next, we show how to construct, for each subterm s of t, the appropriate integer D_s. First, if $s = \overline{r}$ for some $r \in \mathbb{Q}$, then $D_s := d_r$. If $s = x$ for some $x \in \mathrm{VAR}$, then $D_s := 1$. Next, assume we have

defined $D_{s'}$ and $D_{s''}$ for subterms s' and s'' of t. If $s = s' + s''$, then let d be the greatest common divisor of $D_{s'}$ and $D_{s''}$, and $D_s := \frac{D_{s'} D_{s''}}{d}$. If $s = s's''$, then $D_s := D_{s'} D_{s''}$. Finally, if $s = \mathrm{ReLU}(s')$, then $D_s := D_{s'}$.

Let D denote D_t. We start with the term $\overline{D}t$ and modify $\overline{D}t$ into an equivalent term s_0 as follows.

- If $t = \overline{r}$ for some $r \in \mathbb{Q}$, then $r = \frac{n_r}{d_r}$ and $D = d_r$ and thus we set $s_0 := \overline{n_r}$.
- If $t = x$ for some $x \in \mathrm{VAR}$, then $D = 1$ and we set $s_0 := x$.
- If $t = t' + t''$ for some terms t' and t'', then $D = \frac{D_{t'} D_{t''}}{d}$ where d is the greatest common divisor of $D_{t'}$ and $D_{t''}$. We set $s_0 := \overline{\frac{D_{t''}}{d}}\,\overline{D_{t'}}t' + \overline{\frac{D_{t'}}{d}}\,\overline{D_{t''}}t''$.
- If $t = t't''$ for some terms t' and t'', then $D = D_{t'} D_{t''}$ and we set $s_0 := (\overline{D_{t'}}t')(\overline{D_{t''}}t'')$.
- If $t = \mathrm{ReLU}(t')$ for some term t', then $D = D_{t'}$ and we set $s_0 := \mathrm{ReLU}(\overline{D_{t'}}t')$.

In each of the above cases, it is straightforward to show that s_0 is indeed equivalent to $\overline{D}t$. Now in the case of the two first bullets above, we are done. In the case of the last three bullets, we move into the subterm $\overline{D_{t'}}t'$ (and $\overline{D_{t''}}t''$) of s_0 and perform the same transformation to obtain a term s_1 equivalent to s_0. Recursively, we thus obtain s_2, s_3, etc. The process only branches in the case of the third and fourth bullets, and in each such case, the further transformations in one branch do not affect the further transformations in the other. When this process reaches the atomic level, we will have the term s that we seek. A simple induction shows that s is equivalent to $\overline{D}t$, contains only integers and has the same degree as t. □

In particular, due to Lemma 2.2 the above lemma means that we can translate a proto-neuron into a proto-neuron that contains only integers. The next lemma shows that such a proto-neuron s can be translated into RPL and $L\Pi\frac{1}{2}(\odot^-, \to_P^-)$.

Lemma F.6. *Let $\mathcal{L}$ be* RPL *or $L\Pi\frac{1}{2}(\odot^-, \to_P^-)$, and let t be a proto-neuron that contains only integers. For each $i \in \mathbb{R}_+$, there exists some $K \in \mathbb{Z}_+$ such that for each rational number $k \geq K$ we can construct an (i, k)-equivalent formula φ_t of $\mathcal{L}$.*

Proof. Our strategy goes as follows. Because t is a proto-neuron, one of the multiplicands in each multiplication does not contain any variables (and thus gets the same value in every evaluation map). Moreover, t contains only integers. Because of these two facts, we can replace $\odot^k$ in the proof of Lemma E.1 with the iterated sum operator $\bigodot_n'$ defined in Appendix D. We cover the case where $\mathcal{L}$ is RPL as the case where $\mathcal{L}$ is $L\Pi\frac{1}{2}(\odot^-, \to_P^-)$ follows by Proposition 3.9.

Let $i \in \mathbb{R}_+$, define $r_{\max}$, j and K as in the proof of Lemma E.1 and let $k \geq K$ be a rational number. We construct a formula φ_s for each subterm s of t as in the proof of Lemma E.1 with the exception of multiplications, where we define instead:

- If $s = t't''$ for some some proto-neurons t' and t'', then $\deg(t') = 0$ or $\deg(t'') = 0$.
  - If $\deg(t') = \deg(t'') = 0$, then there exists some $z \in \mathbb{Z}$ such that $E(s) = z$ for each evaluation map E. By the choice of k, we have $z \in [-k, k]$ meaning that $\mathrm{scale}_k(z) \in [0, 1]$, and because $k, z \in \mathbb{Q}$, we also have $\mathrm{scale}_k(z) = \frac{k+z}{2k} \in \mathbb{Q}$ and thus $\mathrm{scale}_k(z) \in \mathbb{Q} \cap [0, 1]$. We define $\varphi_s := \overline{\mathrm{scale}_k(z)}$.
  - If $\deg(t') \neq 0$ or $\deg(t'') \neq 0$, then by symmetry and without loss of generality, we may assume that $\deg(t'') = 0$. Then there exists some $z \in \mathbb{Z}$ such that $E(t'') = z$ for each evaluation map E. We define

$$\varphi_s := \begin{cases} \bigodot'_z \varphi_{t'} & \text{if } z \in \mathbb{N} \\ \neg_L \bigodot'_{-z} \varphi_{t'} & \text{if } z \in \mathbb{Z} \setminus \mathbb{N}. \end{cases}$$

Before proving correctness, we note that in Lemma E.1 we had to assume that k is an integer because the connective $\odot^k$ used in the translation is only defined for integers. Here we can relax this assumption because the operator $\bigodot'_z$ does not depend on k, but we still have to assume that k is rational to ensure that the truth constants used in the translation are defined in RPL.

We modify the proof of the correctness of the construction from the proof of Lemma E.1 as follows. Let s be a subterm of t, let E be an evaluation map such that $E(x) \in [-i, i]$ for each $x \in \mathrm{VAR}$ and let V be the unique valuation such that $V(p) = \mathrm{scale}_k(E(x_p))$ for each $p \in \mathrm{PROP}$. We have to show that $V(\varphi_s) = \mathrm{scale}_k(E(s))$. The proof is again by induction over the structure of s and otherwise identical to the proof of Lemma E.1 except for multiplications where we do the following instead. Assume the claim holds for the proto-neurons t' and t'', i.e., $V(\varphi_{t'}) = \mathrm{scale}_k(E(t'))$ and $V(\varphi_{t''}) = \mathrm{scale}_k(E(t''))$.

Assume $s = t't''$. We must show that $V(\varphi_s) = \mathrm{scale}_k(E(s))$. If $\deg(t') = \deg(t'') = 0$, then $\varphi_s = \overline{\mathrm{scale}_k(E(s))}$ and thus $V(\varphi_s) = \mathrm{scale}_k(E(s))$. Assume next that $\deg(t') \neq 0$ or $\deg(t'') \neq 0$. By symmetry, we again assume that $\deg(t'') = 0$ and let $z := E(t'')$ whereby $z \in \mathbb{Z}$. We begin by showing that $E(t') \in \left[-\frac{k}{|z|+1}, \frac{k}{|z|+1}\right]$; by Lemma D.4, this will imply that $V(\bigodot'_{|z|} \varphi_{t'}) = \mathrm{scale}_k(|z|E(t'))$.

We recall from the proof of Lemma E.1 that $|E(t')| \leq \sqrt{k} = \frac{k}{\sqrt{k}}$. Because $k \geq K \geq j^{2\mathrm{length}(t)}$, we have $\sqrt{k} \geq \sqrt{j^{2\mathrm{length}(t)}} = j^{\mathrm{length}(t)}$ and thus $|E(t')| \leq \frac{k}{j^{\mathrm{length}(t)}}$. Because $j \geq 2$ and $\mathrm{length}(t) > \mathrm{length}(t')$, we have $j^{\mathrm{length}(t)} \geq j^{\mathrm{length}(t')} + 1$ and thus $|E(t')| \leq \frac{k}{j^{\mathrm{length}(t')}+1}$. Lastly, we recall from the proof of Lemma E.1 that $|z| = |E(t')| \leq j^{\mathrm{length}(t')}$ and thus $|E(t')| \leq \frac{k}{|z|+1}$, i.e., $E(t') \in \left[-\frac{k}{|z|+1}, \frac{k}{|z|+1}\right]$. Now by Lemma D.4, we have $V(\bigodot'_{|z|} \varphi_{t'}) = \mathrm{scale}_k(|z|E(t'))$. We are now ready to cover the two cases.

- If $z \in \mathbb{N}$, then $|z| = z$ and thus by the above observation we get

$$V(\varphi_s) = V(\bigodot'_z \varphi_{t'}) = \mathrm{scale}_k(zE(t')) = \mathrm{scale}_k(E(s)).$$

- If $z \in \mathbb{Z} \setminus \mathbb{N}$, then $|z| = -z$. By the above, we get $V(\bigodot'_{-z} \varphi_{t'}) = \mathrm{scale}_k(-zE(t'))$. Now by Lemma F.4, we get

$$V(\varphi_s) = V(\neg_L \bigodot'_{-z} \varphi_{t'}) = \mathrm{scale}_k(-(-zE(t'))) = \mathrm{scale}_k(zE(t')) = \mathrm{scale}_k(E(s)).$$

This concludes the proof of the correctness of the construction. □

Example F.7. Consider the neural network $N = (o)$ with the single output neuron

$$o = \mathrm{ReLU}\left(\overline{-8} \cdot \mathrm{ReLU}\left(\overline{4}x + \overline{8}y + \overline{-5}\right) + \overline{2} \cdot \mathrm{ReLU}(\overline{9}x + \overline{7}y + \overline{3}) + \overline{5}\right).$$

An illustration of N is given in Figure 5. Let $i \in \mathbb{R}_+$ and let k be a sufficiently large rational number. By the construction from Lemma F.6, we obtain the following formula of RPL that is (i, k)-equivalent to the neuron o:

$$\begin{aligned} \varphi_o :=& \Big(\Big(\neg_L \bigodot'_8 \Big(\Big(\Big(\bigodot'_4 p_x \oplus' \bigodot'_8 p_y \oplus' \overline{\mathrm{scale}_k(-5)}\Big) \ominus \overline{\tfrac{1}{2}}\Big) \oplus \overline{\tfrac{1}{2}}\Big) \\ & \oplus' \bigodot'_2 \Big(\Big(\Big(\bigodot'_9 p_x \oplus' \bigodot'_7 p_y \oplus' \overline{\mathrm{scale}_k(3)}\Big) \ominus \overline{\tfrac{1}{2}}\Big) \oplus \overline{\tfrac{1}{2}}\Big) \oplus' \overline{\mathrm{scale}_k(5)}\Big) \ominus \overline{\tfrac{1}{2}}\Big) \oplus \overline{\tfrac{1}{2}}. \end{aligned}$$

Now we can combine Lemmas F.5 and F.6 to obtain a translation from proto-neurons to formulae of RPL and $L\Pi\frac{1}{2}(\odot^-, \to_P^-)$.

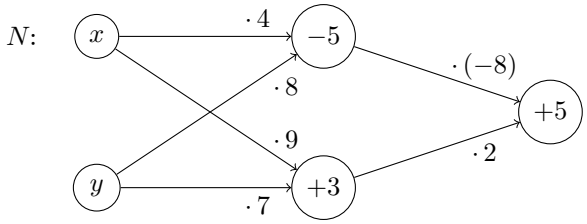

Figure 5: An illustration of the neural network N. Weights are written along the edges and biases are written inside the neurons.

Lemma F.8. *Let $\mathcal{L}$ be* RPL *or $L\Pi\frac{1}{2}(\odot^-, \to_P^-)$. For each proto-neuron and each $i \in \mathbb{R}_+$, there exists some $K \in \mathbb{Z}_+$ such that for each rational number $k \geq K$ we can construct an (i,k)-equivalent formula of $\mathcal{L}$.*

Proof. Let t be a proto-neuron. By Lemma F.5 there exists some $D \in \mathbb{Z}_+$ such that we can construct a proto-neuron s equivalent to $\overline{D}t$ such that s contains only integers. Then by Lemma F.6, for each $i \in \mathbb{R}_+$ there exists some $L_i \in \mathbb{Z}_+$ such that for each rational number $\ell \geq L_i$ we can construct an (i,ℓ)-equivalent formula φ_s of $\mathcal{L}$.

Now let $i \in \mathbb{R}_+$, let $K := \left\lceil \frac{L_i}{D} \right\rceil$ and let $k \geq K$ be a rational number. Now $kD \geq L_i$ and we can construct a formula φ_s of $\mathcal{L}$ that is (i,kD)-equivalent to s. It remains to show that φ_s is (i,k)-equivalent to t. Let E be an evaluation map such that $E(x) \in [-i,i]$ for each $x \in \mathrm{VAR}$, and let V be the unique valuation such that $V(p) = \mathrm{scale}_k(E(x_p))$ for all $p \in \mathrm{PROP}$. We have to show that $V(\varphi_s) = \mathrm{scale}_k(E(t))$. Because φ_s is (i,kD)-equivalent to s, we have $V(\varphi_s) = \mathrm{scale}_{kD}(E(s))$. Because s is equivalent to $\overline{D}t$ we have $E(s) = E(\overline{D}t) = DE(t)$ and thus $V(\varphi_s) = \mathrm{scale}_{kD}(DE(t))$. Now by the definition of scale_k, we get

$$\begin{aligned} V(\varphi_s) &= \mathrm{scale}_{kD}(DE(t)) \\ &= \tfrac{kD+DE(t)}{2kD} \\ &= \tfrac{k+E(t)}{2k} \\ &= \mathrm{scale}_k(E(t)). \end{aligned}$$

Thus, φ_s is (i,k)-equivalent to t.

We note that if we are interested in translating t into a formula of $\mathrm{RPL}(\odot)_{\leq 1}$ or $L\Pi\frac{1}{2}(\to_P^-)_{\leq 1}$ instead, then the application of Lemmas F.5 and F.6 is not necessary (in the case where k is an integer). Instead, by Lemma 2.2, applying Lemma E.1 to t would give a formula of $\mathrm{RPL}(\odot)_{\leq 1}$ or $L\Pi\frac{1}{2}(\to_P^-)_{\leq 1}$. □

Now, we are ready to prove Theorem 4.5.

Theorem 4.5. *Neural networks have the same expressive power (w.r.t. scaling) as the logics* $\mathrm{RPL}(\odot)_{\leq 1}$, $L\Pi\frac{1}{2}(\to_P^-)_{\leq 1}$, RPL *and* $L\Pi\frac{1}{2}(\odot^-, \to_P^-)$.

Proof. Let $\mathcal{L}$ be any of the mentioned logics. Given a neural network $(n_1, \ldots, n_m)$, we can apply the same reasoning as in the proof of Theorem 4.1 to find for each $i \in \mathbb{R}_+$ some $K \in \mathbb{Z}_+$ such that for each rational number $k \geq K$ we can construct a tuple $(\varphi_1, \ldots, \varphi_m)$ of formulae of $\mathcal{L}$ such that φ_j is (i,k)-equivalent to n_j for each $j \in \{1, \ldots, m\}$.

Given a tuple $(\varphi_1, \ldots, \varphi_m)$ of formulae of $\mathcal{L}$, we first use Lemma F.3 to obtain for each φ_i an equivalent neuron n_i. Now, as each n_i can be interpreted as a neural network with a single output neuron, by Lemma F.2 we can replace $n_1, \ldots, n_m$ with neurons $n'_1, \ldots, n'_m$ such that each n'_j is $[0,1]$-equivalent to n_j (and thus equivalent to φ_j), and such that $n'_1, \ldots, n'_m$ all have the same depth d and are identical up to depth $d-1$. This makes $(n'_1, \ldots, n'_m)$ a neural network. □

G. Proof of Corollary 4.8

In this section, we prove Corollary 4.8. We start by recalling the corollary.

Corollary 4.8. $\mathrm{RPL}(\odot)_{\leq 1}$ *and* $L\Pi\frac{1}{2}(\to_P^-)_{\leq 1}$ *have the same expressive power (w.r.t. scaling) as* RPL *and* $L\Pi\frac{1}{2}(\odot^-, \to_P^-)$.

We first establish a useful lemma, connecting scaled equivalence between formulae to scaled and non-scaled equivalence between terms and formulae.

Lemma G.1. *Let* $i, k \in \mathbb{R}_+$ *such that* $1 \leq i \leq k$, *let* φ *and* ψ *each be a formula of* $\mathrm{RPL}(\odot)$ *or* $L\Pi\frac{1}{2}$ *and let* t *be a term of* $\mathbb{Q}[\mathrm{ReLU}, x_i]_{i\in\mathbb{N}}$. *If* φ *and* t *are equivalent and* ψ *is* (i,k)*-equivalent to* t, *then* ψ *is* k*-equivalent to* φ.

Proof. Let V be a valuation and let V' be the valuation such that $V'(p) = \mathrm{scale}_k(V(p))$ for each $p \in \mathrm{PROP}$. Let E be the unique evaluation map such that $E(x) = V(p_x)$ for each $x \in \mathrm{VAR}$, which also implies $E(x) \in [0,1] \subset [-i,i]$ for each $x \in \mathrm{VAR}$ as well as $V'(p) = \mathrm{scale}_k(E(x_p))$ for each $p \in \mathrm{PROP}$. Because ψ is (i,k)-equivalent to t, we have $V'(\psi) = \mathrm{scale}_k(E(t))$, and because φ and t are equivalent, we have $E(t) = V(\varphi)$. Thus $V'(\psi) = \mathrm{scale}_k(V(\varphi))$, i.e., ψ is k-equivalent to φ. □

We are now ready to prove the corollary.

Corollary 4.8. $\mathrm{RPL}(\odot)_{\leq 1}$ *and* $L\Pi\frac{1}{2}(\to_P^-)_{\leq 1}$ *have the same expressive power (w.r.t. scaling) as* RPL *and* $L\Pi\frac{1}{2}(\odot^-, \to_P^-)$.

Proof. Each formula of RPL is a formula of $\mathrm{RPL}(\odot)_{\leq 1}$ and by Proposition 3.7 there exists an equivalent formula of $L\Pi\frac{1}{2}(\to_P^-)_{\leq 1}$. Similarly, each formula of $L\Pi\frac{1}{2}(\odot^-, \to_P^-)$ is a formula of $L\Pi\frac{1}{2}(\to_P^-)_{\leq 1}$ and equivalent to some formula of $\mathrm{RPL}(\odot)_{\leq 1}$. For the converse, by Lemma F.3 each formula φ of $\mathrm{RPL}(\odot)_{\leq 1}$ or $L\Pi\frac{1}{2}(\to_P^-)_{\leq 1}$ is equivalent to a neuron n which in turn by Lemma F.8 is $(1,k)$-equivalent to a formula ψ of RPL or $L\Pi\frac{1}{2}(\odot^-, \to_P^-)$ for some $k > 1$; by Lemma G.1, ψ is k-equivalent to φ. □

H. Translations between neurons and Łukasiewicz logic

In this section, we will discuss the extent to which Łukasiewicz logic can express neural networks. We will first show some interesting places where our previous constructions fail when restricting to Łukasiewicz logic. Then, we will restrict our attention to a subclass of proto-neurons and the fragment of Łukasiewicz logic obtained by omitting the truth constant $\overline{0}$ from the syntax, and show translations back and forth. However, the result is not a fuzzy logic characterisation in the sense of Definition 3.14, because the translation to logic is more restricted than demanded by the definition.

H.1. Applying the previous translations to Łukasiewicz logic

By restricting Lemma F.3 to Łukasiewicz logic, we obtain the following corollary.

Corollary H.1. *For each formula* φ *of Łukasiewicz logic, we can construct an equivalent neuron* n_φ.

Next, we will consider the opposite direction and show some interesting places where the recursive construction from Lemmas E.1 and F.6 fails. The following proposition shows that the construction fails already in the atomic step for almost all rational constants.

Proposition H.2. *Let $i, k \in \mathbb{R}_+$ such that $i \leq k$ and let $r \in \mathbb{Q} \setminus \{-k, k\}$. There exists no formula of Łukasiewicz logic that is (i, k)-equivalent to $\overline{r}$.*

Proof. Theorem 2 in [8] states that for a function $[0,1]^n \to [0,1]$ to be equivalent to a formula of Łukasiewicz logic, it must be representable by a finite number of linear polynomials with integer coefficients. In particular, this holds for constant functions. This implies that there is no formula of Łukasiewicz logic that is equivalent to $\overline{r}$ for any $r \in \mathbb{Q} \cap]0,1[$, as each linear polynomial with integer coefficients can only achieve the value r with finitely many inputs. This naturally holds even if we restrict to only those valuations where the truth values of proposition symbols belong to some subinterval $[a, b]$ of $[0, 1]$ where $a \neq b$, as is the case with (i, k)-equivalence. Thus, for a formula φ to be (i, k)-equivalent to a term $\overline{r}$ for some $r \in \mathbb{Q}$, we must have $\mathrm{scale}_k(r) \in \{0, 1\}$, which implies that $r \in \{-k, k\}$. □

Applying the construction from Lemmas E.1 and F.6 to Łukasiewicz logic also fails in the recursion step. For example, the following proposition shows that even adding variables together is not possible to simulate in Łukasiewicz logic due to the scaling involved.

Proposition H.3. *Let $i, k \in \mathbb{R}_+$ such that $i \leq k$ and let $x, y \in \mathrm{VAR}$. There exists no formula of Łukasiewicz logic that is (i, k)-equivalent to the proto-neuron $x + y$.*

Proof. For the sake of contradiction, assume that φ is a formula of Łukasiewicz logic (i, k)-equivalent to $x + y$. Let E be an evaluation map such that $E(z) \in [-i, i]$ for all $z \in \mathrm{VAR}$ and let V be the valuation such that $V(p) = \mathrm{scale}_k(E(x_p))$ for all $p \in \mathrm{PROP}$. Because φ is (i, k)-equivalent to $x + y$, we must have $V(\varphi) = \mathrm{scale}_k(E(x + y)) = \mathrm{scale}_k(E(x) + E(y))$.

Now, let ψ denote the formula obtained from φ by replacing each instance of p_y with $\neg_L p_x$. Now Lemma F.4 gives us $V(\neg_L p_x) = \mathrm{scale}_k(-E(x)) = \mathrm{scale}_k(E(-x))$. This means that $V(\psi)$ depends on $E(-x)$ in exactly the same way that $V(\varphi)$ depends on $E(y)$, i.e.,

$$V(\psi) = \mathrm{scale}_k(E(x) + E(-x)) = \mathrm{scale}_k(E(x) - E(x)) = \mathrm{scale}_k(0) = \mathrm{scale}_k\left(E\left(\overline{0}\right)\right).$$

This is a contradiction, because ψ is now a formula of Łukasiewicz logic that is (i, k)-equivalent to $\overline{0}$, which is not possible by Proposition H.2. □

Applying the construction from Lemmas E.1 and F.6 to Łukasiewicz logic also fails when we try to apply the ReLU-function to a simple variable, again due to the scaling involved.

Proposition H.4. *Let $i, k \in \mathbb{R}_+$ such that $i \leq k$. There exists no formula of Łukasiewicz logic that is (i, k)-equivalent to the proto-neuron $\mathrm{ReLU}(x)$.*

Proof. The proof is similar to the proof of Proposition H.3. For the sake of contradiction, let φ be a formula of Łukasiewicz logic that is (i, k)-equivalent to the proto-neuron $\mathrm{ReLU}(x)$. Let E be an evaluation map such that $E(y) \in [-i, i]$ for all $y \in \mathrm{VAR}$ and let V be the valuation such that $V(p) = \mathrm{scale}_k(E(x_p))$ for all $p \in \mathrm{PROP}$. Because φ is (i, k)-equivalent to $\mathrm{ReLU}(x)$, we have $V(\varphi) = \mathrm{scale}_k(E(\mathrm{ReLU}(x))) = \mathrm{scale}_k(\max\{0, E(x)\})$.

Now let ψ be the formula obtained from φ by simultaneously replacing each instance of p_x with $\neg_L \varphi$. By Lemma F.4, $V(\neg_L \varphi) = \mathrm{scale}_k(-E(\mathrm{ReLU}(x))) = \mathrm{scale}_k(E(-\mathrm{ReLU}(x)))$. Therefore, $V(\psi)$ depends on $E(-\mathrm{ReLU}(x))$ in the exact same way as $V(\varphi)$ depends on $E(x)$, that is,

$$V(\psi) = \mathrm{scale}_k(\max\{0, E(-\mathrm{ReLU}(x))\}).$$

By the semantics of $\mathbb{Q}[\mathrm{ReLU}, x_i]_{i\in\mathbb{N}}$ and since $-\max\{a, b\} = \min\{-a, -b\}$ for all $a, b \in \mathbb{R}$, we get $E(-\mathrm{ReLU}(x)) = -E(\mathrm{ReLU}(x)) = -\max\{0, E(x)\} = \min\{0, -E(x)\}$, and therefore we have $V(\psi) = \mathrm{scale}_k(\max\{0, \min\{0, -E(x)\}\})$. Now because $\max\{0, \min\{0, a\}\} = 0$ for all $a \in \mathbb{R}$, we have $V(\psi) = \mathrm{scale}_k(0) = \mathrm{scale}_k\left(E\left(\overline{0}\right)\right)$. This is a contradiction, because now ψ is (i, k)-equivalent to $\overline{0}$, which is not possible by Proposition H.2. □

H.2. Łukasiewicz logic without any truth constants

In spite of the fact that the recursive construction from Lemmas E.1 and F.6 fails for Łukasiewicz logic, we can nevertheless reverse-engineer which proto-neurons can be expressed in Łukasiewicz logic.

Let $k \in \mathbb{Q}_+$.[4] **Łukasiewicz proto-neurons over** k are defined as follows.

- Each $x \in \mathrm{VAR}$ is a Łukasiewicz proto-neuron over k.
- If t and t' are Łukasiewicz proto-neurons over k, then so is $\overline{k} - \mathrm{ReLU}(t - t')$.

Proposition H.5. *Let $k \in \mathbb{Q}_+$. For each Łukasiewicz proto-neuron over k and each $i \in \mathbb{R}_+$ such that $i \leq k$, we can construct an (i, k)-equivalent formula of Łukasiewicz logic that does not contain the truth constant $\overline{0}$.*

Proof. We construct the formula recursively over the structure of the Łukasiewicz proto-neuron.

Let E be an evaluation map such that $E(x) \in [-i, i]$ for each $x \in \mathrm{VAR}$ and let V be the valuation such that $V(p) = \mathrm{scale}_k(E(x_p))$ for each $p \in \mathrm{PROP}$. Let t be a Łukasiewicz proto-neuron over k. We construct the (i, k)-equivalent formula φ_t by recursion over the structure of t.

- If $t = x$ for some $x \in \mathrm{VAR}$, then $\varphi_t := p_x$.
- If $t = \overline{k} - \mathrm{ReLU}(t' - t'')$ for some Łukasiewicz proto-neurons t' and t'' over k, then we define $\varphi_t := \varphi_{t'} \to_L \varphi_{t''}$.

Next, we prove the correctness of the translation. The atomic case where $t = x$ for some $x \in \mathrm{VAR}$ is trivial, so assume that the claim holds for Łukasiewicz proto-neurons t' and t'' over k and let $t = \overline{k} - \mathrm{ReLU}(t' - t'')$. It remains to show that $V(\varphi_t) = \mathrm{scale}_k(E(t))$.

By the semantics of $\to_L$, we have $V(\varphi_t) = \min\{1, 1 - V(\varphi_{t'}) + V(\varphi_{t''})\}$. Because $\varphi_{t'}$ and $\varphi_{t''}$ are (i, k)-equivalent to t' and t'', we have $V(\varphi_t) = \min\{1, 1 - \mathrm{scale}_k(E(t')) + \mathrm{scale}_k(E(t''))\}$. By the definition of scale_k, we get

$$\begin{aligned} V(\varphi_t) &= \min\left\{1, 1 - \tfrac{k+E(t')}{2k} + \tfrac{k+E(t'')}{2k}\right\} \\ &= \min\left\{\tfrac{2k}{2k}, \tfrac{2k}{2k} + \tfrac{-k-E(t')}{2k} + \tfrac{k+E(t'')}{2k}\right\} \\ &= \min\left\{\tfrac{k+k}{2k}, \tfrac{k+(k-E(t')+E(t''))}{2k}\right\} \\ &= \min\{\mathrm{scale}_k(k), \mathrm{scale}_k(k - E(t') + E(t''))\}. \end{aligned}$$

[4] We assume that k is rational so that the constructed proto-neurons have rational constant symbols as in the most of the paper, but the definitions and results from here on extend in a natural way to the case where $k \in \mathbb{R}_+$.

Now because scale_k is an increasing function, $\min\{\mathrm{scale}_k(a), \mathrm{scale}_k(b)\} = \mathrm{scale}_k(\min\{a, b\})$ for all $a, b \in \mathbb{R}$ and thus $V(\varphi_t) = \mathrm{scale}_k(\min\{k, k - E(t') + E(t'')\})$. Moreover, because for all $a, b, c \in \mathbb{R}$ we have $\min\{a + b, a + c\} = a + \min\{b, c\}$, we get $V(\varphi_t) = \mathrm{scale}_k(k + \min\{0, -E(t') + E(t'')\})$. Since $\min\{-a, -b\} = -\max\{a, b\}$ for all $a, b \in \mathbb{R}$, we get $V(\varphi_t) = \mathrm{scale}_k(k - \max\{0, E(t') - E(t'')\})$. Now because of the semantics of ReLU, we have $E(t) = k - \max\{0, E(t') - E(t'')\}$ and thus $V(\varphi_t) = \mathrm{scale}_k(E(t))$. This means that φ_t is (i, k)-equivalent to t. □

For Łukasiewicz proto-neurons over 1, we also obtain a reverse translation, expressed below.

Proposition H.6. *For each formula of Łukasiewicz logic that does not contain the truth constant $\overline{0}$, we can construct an equivalent Łukasiewicz proto-neuron over* 1.

Proof. This follows directly from the proof of Lemma E.2, when we restrict to formulae of Łukasiewicz logic that do not contain the truth constant $\overline{0}$, which are simply built from proposition symbols and the connective $\to_L$. □

Propositions H.5 and H.6 together constitute a fuzzy logic characterisation of Łukasiewicz proto-neurons over 1 via the fragment $\mathcal{L}$ of Łukasiewicz logic obtained by omitting $\overline{0}$ from the grammar. However, it is not a characterisation in the sense of Definition 3.14, since the definition requires that we obtain an (i, k)-equivalent formula for each $i \in \mathbb{R}_+$ while Proposition H.5 only holds when $i \leq k$ and we have $k = 1$. It also remains an open question for which Łukasiewicz proto-neurons over 1 we can obtain an equivalent neuron.

Lastly, by Propositions H.5 and H.6, we obtain the following perhaps amusing corollary.

Corollary H.7. *Each formula of Łukasiewicz logic that does not contain the truth constant $\overline{0}$ is 1-equivalent to itself.*

Proof. Let φ be a formula of Łukasiewicz logic that does not contain the truth constant $\overline{0}$. By Proposition H.6, we can construct a Łukasiewicz proto-neuron t over 1 that is equivalent to φ, and by Proposition H.5, we can construct a formula φ' of Łukasiewicz logic that does not contain the truth constant $\overline{0}$ such that φ' is $(1, 1)$-equivalent to t. By Lemma G.1, φ' is 1-equivalent to φ.

It remains to show that $\varphi' = \varphi$ when we construct t and φ' according to the proofs of Propositions H.5 and H.6. We prove the claim by induction over the structure of φ. If $\varphi = p$ for some $p \in \mathrm{PROP}$, then by the construction from the proof of Lemma E.2, we have $t = x_p$, and by the construction in the proof of Proposition H.5, we get $\varphi' = p = \varphi$. Next, assume the claim holds for formulae ψ and θ, let t_ψ and t_θ denote the proto-neurons obtained by the construction in the proof of Lemma E.2, and let ψ' and θ' denote the formulae obtained by the construction in the proof of Proposition H.5. Let $\varphi = \psi \to_L \theta$. Now, by the construction in Lemma E.2, we get $t = \overline{1} - \mathrm{ReLU}(t_\psi - t_\theta)$. By the construction in the proof of Proposition H.5, we have $\varphi' = \psi' \to_L \theta'$. By the induction hypothesis, we have $\psi' = \psi$ and $\theta' = \theta$ and thus $\varphi' = \psi \to_L \theta = \varphi$. □

Corollary H.7 shows that each formula of Łukasiewicz logic that does not contain the truth constant $\overline{0}$ can use the interval $\left[\frac{1}{2}, 1\right]$ to simulate its own behaviour in the interval $[0, 1]$.