Machine Learning (2023) 112:3293–3331
https://doi.org/10.1007/s10994-023-06310-3

Refining neural network predictions using background knowledge

Alessandro Daniele[1] · **Emile van Krieken**[2] · **Luciano Serafini**[1] · **Frank van Harmelen**[2]

Received: 8 June 2022 / Revised: 23 January 2023 / Accepted: 27 January 2023 /
Published online: 14 March 2023

Abstract
Recent work has shown learning systems can use logical background knowledge to compensate for a lack of labeled training data. Many methods work by creating a loss function that encodes this knowledge. However, often the logic is discarded after training, even if it is still helpful at test time. Instead, we ensure neural network predictions satisfy the knowledge by refining the predictions with an extra computation step. We introduce differentiable *refinement functions* that find a corrected prediction close to the original prediction. We study how to effectively and efficiently compute these refinement functions. Using a new algorithm called iterative local refinement (ILR), we combine refinement functions to find refined predictions for logical formulas of any complexity. ILR finds refinements on complex SAT formulas in significantly fewer iterations and frequently finds solutions where gradient descent can not. Finally, ILR produces competitive results in the MNIST addition task.

Keywords Neurosymbolic AI · Fuzzy logic · Optimization

Alessandro Daniele and Emile van Krieken have contributed equally to this work.

Editors: Alireza Tamaddoni-Nezhad, Alan Bundy, Luc De Raedt, Artur d'Avila Garcez, Sebastijan Dumancić, Cèsar Ferri, Pascal Hitzler, Nikos Katzouris, Denis Mareschal, Stephen Muggleton, Ute Schmid.

✉ Emile van Krieken
e.van.krieken@vu.nl

Alessandro Daniele
daniele@fbk.eu

[1] Data and Knowledge Management unit, Fondazione Bruno Kessler, via Sommarive 18, 38123 Trento, Italy

[2] Department of Computer Science, Vrije Universiteit Amsterdam, de Boelelaan 1081a, 1081HV Amsterdam, Netherlands

1 Introduction

Recent years have shown promising examples of using symbolic background knowledge in learning systems: From training classifiers with weak supervision signals (Manhaeve et al., 2018), generalizing learned classifiers to new tasks (Roychowdhury et al., 2021), compensating for a lack of good supervised data (Diligenti et al., 2017; Donadello et al., 2017), to enforcing the structure of outputs through a logical specification (Xu et al., 2018). The main idea underlying these integrations of learning and reasoning, often called neurosymbolic integration, is that background knowledge can complement the neural network when one lacks high-quality labeled data (Giunchiglia et al., 2022). Although pure deep learning approaches excel when learning over *vast* quantities of data with *gigantic* amounts of compute (Chowdhery et al., 2022; Ramesh et al., 2022), we cannot afford this luxury for most tasks.

Many neurosymbolic methods work by creating a differentiable loss function that encodes the background knowledge (Fig. 1a). However, often the logic is discarded after training, even though this background knowledge could still be helpful at test time (Roychowdhury et al., 2021; Giunchiglia et al., 2022a). Instead, we ensure we constrain the neural network with the background knowledge, both during train time and test time, by correcting its output to satisfy the background knowledge (Fig. 1b). In particular, we consider how to make such corrections while being as close as possible to the original predictions of the neural network.

We study how to effectively and efficiently correct the neural network by ensuring its predictions satisfy the symbolic background knowledge. In particular, we consider fuzzy logics formed using functions called t-norms (Klement et al., 2000; Ross, 2010). Prior work has shown how to use a gradient ascent-based optimization procedure to find a prediction that satisfies this fuzzy background knowledge (Diligenti et al., 2017; Roychowdhury et al., 2021). However, a recent model called KENN (Clarke et al., 1993; Daniele & Serafini, 2019) shows how to compute the correction analytically for a fragment of the Gödel logic.

To extend this line of work, we introduce the concept of *refinement functions* and derive refinement functions for many fuzzy logic operators. Refinement functions are functions that find a prediction that satisfies the background knowledge while staying close to the neural network's original prediction. Using a new algorithm called *Iterative Local Refinement* (ILR), we can combine refinement functions for different fuzzy logic operators to efficiently find refinements for logical formulas of any complexity. Since

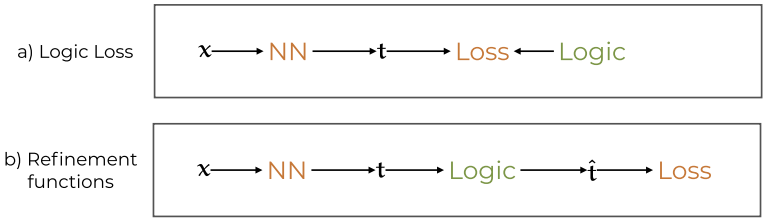

Fig. 1 Comparing different approaches for constraining neural networks with background knowledge. Loss-based approaches include LTN, SBR, and Semantic Loss, while KENN, CCN(h), and SBR-CC are representatives for refinement functions. $\mathbf{x}$ represents a high-dimensional input for a neural network, $\boldsymbol{t}$ represents the initial predictions of this neural network and $\hat{\boldsymbol{t}}$ represents the refined prediction that incorporates the background knowledge

refinement functions are differentiable, we can easily integrate them as a neural network layer. In our experiments, we compare ILR with an approach using gradient ascent. We find that ILR finds optimal refinements in significantly fewer iterations. Moreover, ILR often produces results that stay closer to the original predictions or better satisfy the background knowledge. Finally, we evaluate ILR on the MNIST Addition task (Manhaeve et al., 2018) and show how to combine ILR with neural networks to solve neurosymbolic tasks.

In summary, our contributions are:

1. We formalize the concept of minimal refinement functions in Sect. 4.
2. We introduce the ILR algorithm in Sect. 5, which uses the minimal refinement functions for individual fuzzy operators to find refinements for general logical formulas.
3. We discuss how to use ILR for neurosymbolic AI in Sect. 6, where we exploit the fact that ILR is a differentiable algorithm.
4. We analytically derive minimal refinement functions for individual fuzzy operators constructed from the Gödel, Łukasiewicz, and product t-norms in Sect. 7.2.
5. We discuss a large class of t-norms for which we can analytically derive minimal refinement functions in Sect. 7.
6. We compare ILR to gradient descent approaches and show it finds refinements on complex SAT formulas in significantly fewer iterations and frequently finds solutions where gradient descent can not.
7. We apply ILR to the MNIST Addition task (Manhaeve et al., 2018) to test how ILR behaves when injecting knowledge into neural network models.

2 Related work

ILR falls into a larger body of work that attempts to integrate background knowledge expressed as logical formulas into neural networks. For an overview, see (Giunchiglia et al., 2022). Figure 1 shows two categories that most methods fall in. The first only use background knowledge during training in the form of a loss function (Badreddine et al., 2022; Xu et al., 2018; Diligenti et al., 2017; Fischer et al., 2019; Yang et al., 2022; van Krieken et al., 2022). The second considers the background knowledge as part of the model and enforces the knowledge at test time (Daniele & Serafini, 2019; Wang et al., 2019; Giunchiglia & Lukasiewicz, 2021; Ahmed et al., 2022; Hoernle et al., 2022; Dragone et al., 2021). ILR is a method in the second category. We note that these approaches can be combined (Giunchiglia et al., 2022a; Roychowdhury et al., 2021).

First, we discuss approaches that construct loss functions from the logical formulas (Fig. 1a). These loss functions measure when the deep learning model violates the background knowledge, such that minimizing the loss function amounts to "correcting" such violations (van Krieken et al., 2022). While these methods show significant empirical improvement, they do not guarantee that the neural network will satisfy the formulas outside the training data. LTN and SBR (Badreddine et al., 2022; Diligenti et al., 2017) use fuzzy logic to provide compatibility with neural network learning, while Semantic Loss (Xu et al., 2018) uses probabilistic logics. It is possible to extend the formalization of refinement functions to probabilistic logics by defining a suitable notion of minimality. One example is the KL-divergence between the original and refined distributions over ground atoms.

Among the methods where knowledge is part of the model, KENN inspired ILR (Daniele & Serafini, 2019, 2022). KENN is a framework that injects knowledge into neural networks by iteratively refining its predictions. It uses a relaxed version of the Gödel t-conorm obtained through a relaxation of the argmax function, which it applies in logit space. Closely related to both ILR and KENN is CCN(h) (Giunchiglia & Lukasiewicz, 2021), which we see as computing the minimal refinement function for stratified normal logic programs under Gödel t-norm semantics. We discuss this connection in more detail in Sect. 7.2.1.

The loss-function-based method SBR also introduces a procedure for using the logical formulas at test time in the context of collective classification (Diligenti et al., 2017; Roychowdhury et al., 2021). Unlike KENN (Daniele & Serafini, 2019), these approaches do not enforce the background knowledge during training but only use it as a test time procedure. In particular, (Roychowdhury et al., 2021) shows that doing these corrections at test time improves upon just using the loss-function approach. Unlike our analytic approach to refinement functions, SBR finds new predictions using a gradient descent procedure very similar to the algorithm we discuss in Sect. 9.1.2. We show it is much slower to compute than ILR.

Another method closely related to ILR is the neural network layer SATNet (Wang et al., 2019), which has a setup closely related to ours. However, SATNet does not have a notion of minimality and uses a different underlying logic constructed from a semidefinite relaxation. DeepProbLog (Manhaeve et al., 2018) also is a probabilistic logic, but unlike Semantic Loss is used to derive new statements through proofs and cannot directly be used to correct the neural network on predictions that do not satisfy the background knowledge. Instead, ILR can be used to inject constraints on the output of a neural network, and to prove new statements starting from the neural network predictions.

Finally, some methods are limited to equality and inequality constraints rather than general symbolic background knowledge (Fischer et al., 2019; Hoernle et al., 2022). DL2 (Fischer et al., 2019) combines these constraints into a real-valued loss function, while MultiplexNet (Hoernle et al., 2022) adds the knowledge as part of the model. However, MultiplexNet requires expressing the logical formulas as a DNF formula, which is hard to scale.

3 Fuzzy operators

We will first provide the necessary background knowledge for defining and analyzing minimal refinement functions. In particular, we will consider fuzzy operators, which generalize the connectives of classical boolean logic. For formal treatments of the study of fuzzy operators, we refer the reader to (Klement et al., 2000), which discusses t-norms and t-conorms, to Jayaram and Baczynski (2008) for fuzzy implications, to Calvo et al. (2002) for aggregation functions, and to van Krieken et al. (2022) for an analysis of the derivatives of these operators.

Definition 1 A function $T : [0,1]^2 \to [0,1]$ is a *t-norm* (triangular norm) if it is commutative, associative, increasing in both arguments, and if for all $t \in [0,1]$, $T(1,t) = t$.

Similarly, a function $S : [0,1]^2 \to [0,1]$ is a *t-conorm* if the last condition instead is that for all $t \in [0,1]$, $S(0,t) = t$.

Dual t-conorms are formed from a t-norm T using $S(t_1,t_2) = 1 - T(1-t_1, 1-t_2)$. We list the n-arity extensions, constructed using $T(\boldsymbol{t}) = T(t_1, T(\boldsymbol{t}_{2:n}))$, $T(t_i) = t_i$ of three basic t-norms in Table 1. Here $\boldsymbol{t} = [t_1, ..., t_n]^\top \in [0,1]^n$ is a vector of fuzzy truth values, which

Table 1 Some common t-norms extended to n-arity aggregation operators

Name	T-norm
Minimum	$T_G(\boldsymbol{t}) = \min_{i=1}^{n} t_i$
Product	$T_P(\boldsymbol{t}) = \prod_{i=1}^{n} t_i$
Łukasiewicz	$T_L(\boldsymbol{t}) = \max(\sum_{i=1}^{n} t_i - (n-1), 0)$

we will often refer to as *(truth) vectors*. These n-arity extensions are examples of fuzzy aggregation operators (Calvo et al., 2002).

Definition 2 A function $I : [0,1]^2 \rightarrow [0,1]$ is a *fuzzy implication* if for all $t_1, t_2 \in [0,1]$, $I(\cdot, t_2)$ is decreasing, $I(t_1, \cdot)$ is increasing and if $I(0,0) = 1$, $I(1,1) = 1$ and $I(1,0) = 0$.

Note that fuzzy implications do not have n-ary extensions as they are not associative. The so-called *S-implications* are formed from the t-conorm by generalizing the material implication using $I(a,c) = S(1-a,c)$. Furthermore, every t-norm induces a unique *residuum* or *R-implication* (Jayaram & Baczynski, 2008) $R_T(a,c) = \sup\{z | T(z,a) \leq c\}$.

Logical formulas φ can be evaluated using compositions of fuzzy operators. We assume φ is a propositional logic formula, but we note the evaluation procedure can be extended to grounded first-order logical formulas on finite domains. For instance, (Daniele & Serafini, 2022) introduced a technique for propositionalizing universally quantified formulas of predicate logic in the context of KENN. Moreover, this technique can be extended to existential quantification by treating it as a disjunction. We assume a set of propositions $\mathcal{P} = \{P_1, ..., P_n\}$ and constants $\mathcal{C} = \{C_1, ..., C_m\}$, where each constant has a fixed value $C_i \in [0,1]$.

Definition 3 If T is a t-norm, S a t-conorm and I a fuzzy implication, then the *fuzzy evaluation operator* $f_\varphi : [0,1]^n \rightarrow [0,1]$ of the formula φ with propositions $\mathcal{P}$ and constants $\mathcal{C}$ is a function of truth vectors $\boldsymbol{t}$ and given as

$$f_{P_i}(\boldsymbol{t}) = t_i \tag{1}$$

$$f_{C_j}(\boldsymbol{t}) = C_j \tag{2}$$

$$f_{\neg\phi}(\boldsymbol{t}) = 1 - f_\phi(\boldsymbol{t}) \tag{3}$$

$$f_{\bigwedge_{j=1}^{m} \phi_j}(\boldsymbol{t}) = T(f_{\phi_1}(\boldsymbol{t}), ..., f_{\phi_m}(\boldsymbol{t})) \tag{4}$$

$$f_{\bigvee_{j=1}^{m} \phi_j}(\boldsymbol{t}) = S(f_{\phi_1}(\boldsymbol{t}), ..., f_{\phi_m}(\boldsymbol{t})) \tag{5}$$

$$f_{\phi \rightarrow \psi}(\boldsymbol{t}) = I(f_\phi(\boldsymbol{t}), f_\psi(\boldsymbol{t})), \tag{6}$$

where we match the structure of the formula φ in the subscript f_φ.

4 Minimal fuzzy Refinement functions

We will next define (fuzzy) refinement functions, which consider how to change the input arguments of fuzzy operators such that the output of the operators is a given truth value. refinement functions prefer changes to the input arguments that are as small as possible. We will introduce several definitions to facilitate studying this concept. The first is an optimality criterion.

Definition 4 (Fuzzy refinement function)

Let $f_\varphi : [0,1]^n \to [0,1]$ be a fuzzy evaluation operator. Then $\hat{\boldsymbol{t}} : [0,1]^n$ is called a *refined (truth) vector* for the *refinement value* $\hat{t}_\varphi \in [0,1]$ if $f_\varphi(\hat{\boldsymbol{t}}) = \hat{t}_\varphi$.

Furthermore, let $\min_\varphi = \min_{\hat{\boldsymbol{t}} \in [0,1]^n} f_\varphi(\hat{\boldsymbol{t}})$ and $\max_\varphi = \max_{\hat{\boldsymbol{t}} \in [0,1]^n} f_\varphi(\hat{\boldsymbol{t}})$. Then $\rho : [0,1]^n \times [0,1] \to [0,1]^n$ is a *(fuzzy) refinement function*[1]

for f_φ if for all $\boldsymbol{t} \in [0,1]^n$,

1. for all $\hat{t}_\varphi \in [\min_\varphi, \max_\varphi]$, $\rho(\boldsymbol{t}, \hat{t}_\varphi)$ is a refined vector for $\hat{t}_\varphi$;
2. for all $\hat{t}_\varphi < \min_\varphi$, $\rho(\boldsymbol{t}, \hat{t}_\varphi) = \rho(\boldsymbol{t}, \min_\varphi)$;
3. for all $\hat{t}_\varphi > \max_\varphi$, $\rho(\boldsymbol{t}, \hat{t}_\varphi) = \rho(\boldsymbol{t}, \max_\varphi)$.

A refinement function for f_φ changes the input truth vector in such a way that the new output of f_φ will be $\hat{t}_\varphi$. Whenever $\hat{t}_\varphi$ is high, we want the refined vector to satisfy the formula φ, while if $\hat{t}_\varphi$ is low, we want it to satisfy its negation. When $\hat{t}_\varphi = 1$, the constraint created by the formula is a hard constraint, while if it is in (0, 1), this constraint is soft. We require bounding the set of possible $\hat{t}_\varphi$ by $\min_\varphi$ and $\max_\varphi$ since if there are constants C_i, or if φ has no satisfying (discrete) solutions, there can be formulas such that there can be no refined vectors $\hat{\boldsymbol{t}}$ for which $f_\varphi(\hat{\boldsymbol{t}})$ equals 1.

Next, we introduce a notion of minimality of refinement functions. The intuition behind this concept is that we prefer the new output, the refined vector $\hat{\boldsymbol{t}}$, to stay as close as possible to the original truth vector $\boldsymbol{t}$. Therefore, we assume we want to find a truth vector near the neural network's output that satisfies the background knowledge.

Definition 5 (Minimal refinement function) Let ρ^* be a refinement function for operator f_φ. ρ^* is a *minimal* refinement function with respect to some norm $\|\cdot\|$ if for each $\boldsymbol{t} \in [0,1]^n$ and $\hat{t}_\varphi \in [\min_\varphi, \max_\varphi]$, there is no refined vector $\hat{\boldsymbol{t}}'$ for $\hat{t}_\varphi$ such that $\|\rho^*(\boldsymbol{t}, \hat{t}_\varphi) - \boldsymbol{t}\| > \|\hat{\boldsymbol{t}}' - \boldsymbol{t}\|$.

For a particular fuzzy evaluation operator f_φ, finding the minimal refinement function corresponds to solving the following optimization problem:

$$\begin{aligned} \text{For all} \quad & \boldsymbol{t} \in [0,1]^n, \hat{t}_\varphi \in [\min_\varphi, \max_\varphi] \\ \min_{\hat{\boldsymbol{t}}} \quad & \|\hat{\boldsymbol{t}} - \boldsymbol{t}\| \\ \text{such that} \quad & f_\varphi(\hat{\boldsymbol{t}}) = \hat{t}_\varphi, \\ & 0 \le \hat{t}_i \le 1 \end{aligned} \tag{7}$$

[1] The concept of refinement functions is closely related to the concept of *Fuzzy boost function* in the KENN paper (Daniele & Serafini, 2019).

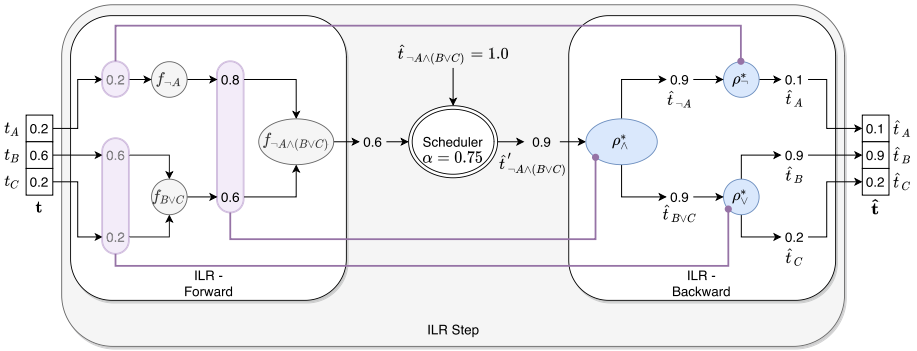

Fig. 2 Visualization of one step of ILR for the Gödel logic and formula $\phi = \neg A \wedge (B \vee C)$. In the forward pass (left), ILR computes the truth value of ϕ. In the backward pass (right), ILR traverses the computational graph of the forward step in reverse to calculate the refined vector $\hat{\boldsymbol{t}}$. ILR substitutes each fuzzy operator of the forward pass with the corresponding refinement function. Each refinement function receives as input the initial truth values used by the fuzzy operator in the forward step (purple lines) and the target value for the corresponding subformula. The scheduler calculates the target value $\hat{t}'_{\neg A \wedge (B \vee C)}$ for the entire formula, which ILR calls between the forward and backward steps (Color figure online)

For some f_φ we can solve this problem analytically using the Karush-Kuhn-Tucker (KKT) conditions. However, while $\|\cdot\|$ is convex, f_φ (usually) is not. Therefore, we can not rely on efficient convex solvers. Furthermore, for strict t-norms, finding exact solutions to this problem is equivalent to solving PMaxSAT when $\hat{t}_\varphi = 1$ (Diligenti et al., 2017; Giunchiglia et al., 2022a), hence this problem is NP-complete. In Sects. 7 and 8, we will derive minimal refinement functions for a large amount of individual fuzzy operators analytically. These results are the theoretical contribution of this paper. We first discuss in Sect. 5 a method called ILR for finding general solutions to the problem of finding minimal refinement functions. ILR uses the analytical minimal refinement functions of individual fuzzy operators in a forward-backward algorithm. Then, in Sect. 6, we discuss how to use this algorithm for neurosymbolic AI.

5 Iterative local refinement

We introduce a fast, iterative, differentiable but approximate algorithm called *Iterative local refinement (ILR)* that finds minimal refinement functions for general formulas. ILR is a forward-backward algorithm acting on the computation graph of formulas. First, it traverses the graph from its leaves to its root to compute the current truth values of subformulas. Then, it traverses the graph back from its root to the leaves to compute new truth values for the subformulas. ILR makes use of analytical minimal refinement functions to perform this backward pass. ILR is a differentiable algorithm if the fuzzy operators and their corresponding minimal refinement functions are differentiable as it computes compositions of these functions.

Algorithm 1 contains the pseudocode of ILR, and Fig. 2 presents an example of a single step (lines 3 to 7 of the algorithm) for the formula $\varphi = \neg A \wedge (B \vee C)$ under the Gödel semantics.

First, ILR computes the truth value of the formula in the forward pass, as shown on the left side of Fig. 2. ILR saves the truth vectors of intermediate subformulas in $\boldsymbol{t}_{\text{sub}}$,

which are presented in Fig. 2 as the numbers inside the purple shapes. Then, ILR calls a scheduler to determine the right target value for the formula φ. The target value is $\hat{t}'_{\varphi} = \alpha \cdot (1 - 0.6) = 0.9$ for our example. The scheduling mechanism smooths the updates ILR makes. We implement this in line 6 of Algorithm 1. It works by choosing a different refined value at each iteration: The difference between the current truth value and the refined value is multiplied by a scheduling parameter α, which we choose to be either 0.1 or 1 (no scheduling). While usually not necessary, for some formulas, the scheduling mechanism allowed for finding better solutions.

Following the scheduler, ILR computes the backward step in rows from 13 to 19 in Algorithm 1. It changes the input truth vector $\boldsymbol{t}$ based on the formula φ. Note that the formula φ in Fig. 2 is a conjunction of two subformulas ($\varphi_1 = \neg A$ and $\varphi_2 = B \vee C$). ILR applies refinement functions recursively by treating the subformulas as literals: We give the truth values of φ_1 and φ_2 we saved in the forward pass, as inputs to the refinement function. In the example, we use the refinement function for the Gödel t-norm.

The refinement function updates the truth values of φ_1 and φ_2. Then, we interpret these new values as the *target* truth values for the formulas φ_1 and φ_2. This allows us to apply the refinement proccedure recursively. For instance, in Fig. 2, the refined truth values $\hat{t}_{\neg A}$ and $\hat{t}_{B \vee C}$ can be interpreted as the target truth values for $\neg A$ and $B \vee C$, respectively. Then, by applying the refinement functions for negation[2] and t-conorm, we can obtain the truth values of A, B and C.

One choice in ILR is how to combine the results from different subformulas. Indeed, when a proposition appears in multiple subformulas, it can be assigned multiple different refined values. As an example, suppose the formula of Fig. 2 was $\varphi = \neg A \wedge (B \vee A)$, with the proposition C replaced by A. While similar to the previous formula, A is repeated twice. Consequently, the algorithm produces two different refined values for A. We found the heuristic in line 18 generally works well, which takes the $\hat{t}_j$ with the largest absolute value. We also explored two other heuristics. In the first, we averaged the different refined values, but this took significantly longer to converge. The second heuristic we explored was the smallest absolute value, which frequently did not find solutions. Another choice is the convergence criterion. A simple option is to stop running the algorithm whenever it has stopped getting closer to the refined value for a couple of iterations. In our experiments, we observed that ILR monotonically decreases the distance to the refined value, after which it gets stuck on a single local optimum or oscillates between two local minima.

ILR is not guaranteed to find a refined vector $\hat{\boldsymbol{t}}$ such that $f_{\varphi}(\hat{\boldsymbol{t}}) = \hat{t}_{\varphi}$. This is easy to see theoretically because, for many fuzzy logics like the product and Gödel logics, $\hat{t}_{\varphi} = 1$ corresponds to the PMaxSAT problem, which is NP-complete (Diligenti et al., 2017; Giunchiglia et al., 2022a), while ILR has linear time complexity. However, this is traded off by 1) being highly efficient, usually requiring only a couple of iterations for convergence, and 2) not having any hyperparameters to tune, except arguably for the combination function. Furthermore, ILR usually converges quickly in neurosymbolic settings since background knowledge is very structured, and the solution space is relatively dense. These settings are unlike the randomly generated SAT problems we study in Sect. 9.1.3. These contain little structure the ILR algorithm can exploit.

[2] Note that the minimal refinement function for the negation is trivial since there is only a feasible solution. For this reason, we omitted it from our analysis.

Algorithm 1 Iterative Local Refinement

Require: $\varphi, \hat{t}_\varphi, \boldsymbol{t}, \alpha \in (0, 1]$
$\boldsymbol{t}' \leftarrow \boldsymbol{t}$
while not converged **do**
  $\boldsymbol{t}_{\mathsf{sub}} \leftarrow \{\}$
  for subformula ϕ of φ **do**
    $\boldsymbol{t}_{\mathsf{sub}}[\phi] \leftarrow f_\phi(\boldsymbol{t}')$ ▷ Forward pass using Definition 3
  $\hat{t}'_\varphi \leftarrow f_\varphi(\boldsymbol{t}) + \alpha \cdot (\hat{t}_\varphi - f_\varphi(\boldsymbol{t}))$
  $\boldsymbol{t}' \leftarrow \textsc{Backward}(\varphi, \hat{t}'_\varphi, \boldsymbol{t}_{\mathsf{sub}})$
return $\boldsymbol{t}'$
function $\textsc{Backward}(P_i, \hat{t}_{P_i}, \boldsymbol{t}_{\mathsf{sub}})$
  return $[t_1, \ldots, \hat{t}_{P_i}, \ldots, t_n]^\top$ ▷ $\boldsymbol{t}$ except at position i.
function $\textsc{Backward}(\neg\phi, \hat{t}_{\neg\phi}, \boldsymbol{t}_{\mathsf{sub}})$
  return $\textsc{Backward}(\phi, 1 - \hat{t}_{\neg\phi}, \boldsymbol{t}_{\mathsf{sub}})$
function $\textsc{Backward}(\bigwedge_{i=1}^m \phi_i, \hat{t}_\varphi, \boldsymbol{t}_{\mathsf{sub}})$
  $\hat{\boldsymbol{t}}_\wedge \leftarrow \rho^*_T([\boldsymbol{t}_{\mathsf{sub}}[\phi_1], ..., \boldsymbol{t}_{\mathsf{sub}}[\phi_m]]^\top, \hat{t}_\varphi)$ ▷ Minimal refinement function
  $\hat{\boldsymbol{t}} \leftarrow \boldsymbol{0}$
  for $i \leftarrow 1$ to m **do**
    $\hat{\boldsymbol{t}}' \leftarrow \textsc{Backward}(\phi_i, \hat{t}_{\wedge,i}, \boldsymbol{t}_{\mathsf{sub}})$
    $\hat{t}_j \leftarrow$ **if** $|\hat{t}_j| > |\hat{t}'_j|$ **then** $\hat{t}_j$ **else** $\hat{t}'_j$ for all $j \in \{1, ..., n\}$
  return $\hat{\boldsymbol{t}}$

6 Neuro-symbolic AI using ILR

The ILR algorithm can be added as a module after a neural network g to create a neuro-symbolic AI model. The neural network predicts (possibly some of) the initial truth values $\boldsymbol{t}$. Since both the forward and backward passes of ILR are differentiable computations, we can treat ILR as a constrained output layer (Giunchiglia et al., 2022). For instance, in Fig. 2, the input $\boldsymbol{t}$ could be generated by the neural network, and we provide supervision directly on the predictions $\hat{\boldsymbol{t}}$. With ILR, the predictions, i.e., the refined vector $\hat{\boldsymbol{t}}$, take the background knowledge into account while staying close to the original predictions made by the neural network. Loss functions like cross-entropy can use $\hat{\boldsymbol{t}}$ as the prediction. We train the neural network g by minimizing the loss function with gradient descent and backpropagating through the ILR layer.

One strength of ILR is the flexibility of the refinement values $\hat{t}_{\varphi_i}$ for each formula φ_i. These can be set to 1 to treat φ_i as a hard constraint that always needs to be satisfied. Alternatively, refinement values can be trained as part of a larger deep learning model. Since ILR is a differentiable layer, we can compute gradients of the refinement values. This procedure allows ILR to learn what formulas are useful for prediction. For instance, in Fig. 2, $\hat{t}_{\neg A \wedge (B \vee C)}$ can either be given or act as a parameter of the model that is learned together with the neural network parameters.

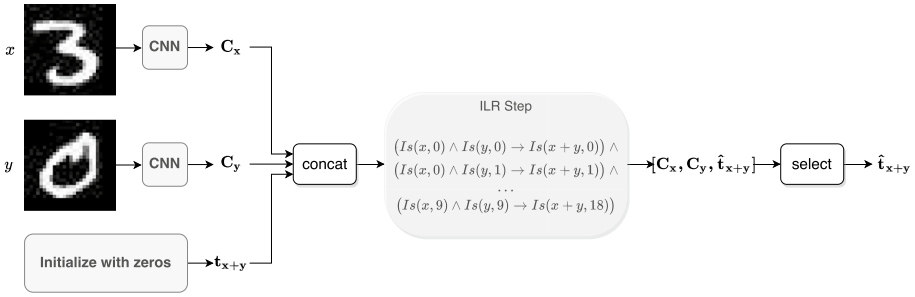

Fig. 3 Neurosymbolic architecture based on ILR for the MNIST Addition task. A CNN takes two images of MNIST digits, returning their classification. The CNN predictions are concatenated with a vector of zeros, representing the initial prediction for the Addition task. We perform an ILR step to update the sum of the two numbers, which is the final output of the model

We give an example of the integration of ILR with a neural network in Fig. 3, where we use ILR for the MNIST Addition task proposed by Manhaeve et al. (2018). In this task, we have access to a training set composed of triplets (x, y, z), where x and y are images of MNIST (LeCun, 2010) handwritten digits, and z is a label representing an integer in the range $\{0, ..., 18\}$, corresponding to the sum of the digits represented by x and y. The task consists of learning the addition function and a classifier for the MNIST digits, with supervision only on the sums. To achieve this, knowledge consisting of the rules of addition is given. For instance, the rule $Is(x, 3) \wedge Is(y, 2) \rightarrow Is(x + y, 5)$ states that the sum of 3 and 2 is 5.

The architecture of the model presented in Fig. 3 consists of a neural network (a CNN) that performs digit recognition on the inputs x and y. After this step, ILR predicts a truth value for each possible sum. Notice that we define the CNN outputs $\boldsymbol{C}_x, \boldsymbol{C}_y \in [0, 1]^{10}$ as constants, i.e., ILR does not change the predictions of the digits. Moreover, the initial prediction for the truth vector of possible sums $\boldsymbol{t}_{x+y} \in [0, 1]^{19}$ is the zero vector. This allows ILR to act as a proof-based method. Indeed, similarly to DeepProbLog (Manhaeve et al., 2018), the architecture proposed in Fig. 3 uses the knowledge in combination with the predictions of the neural network to derive truth values for new statements (the sum of the two digits). We apply the loss function to the final predictions $\hat{\boldsymbol{t}}_{x+y}$. During learning, the error is back-propagated through the entire model, reaching the CNN, which learns to classify the MNIST images from indirect supervision.

We present the results obtained by ILR in Sect. 9.2, and compare its performance with other neurosymbolic AI frameworks.

7 Analytical minimal refinement functions

Having introduced the ILR algorithm, we next study the problem of finding minimal refinement functions for individual fuzzy operators. We need these in closed form to compute the ILR algorithm, as ILR uses them during the backward pass. This section first discusses several transformations of minimal refinement functions and gives the minimal refinement functions of the basic t-norms Gödel, Łukasiewicz and product. In Sect. 8, we investigate a large class of t-norms for which we have closed-form formulas for the minimal refinement functions.

7.1 General results

We first provide several basic results on minimal refinement functions for fuzzy operators. In particular, we will consider formulas such as $\varphi = \bigwedge_{i=1}^{n} P_i \bigwedge_{i=1}^{m} C_i$, that is, conjunctions of propositions and constants. As an abuse of notation, from here on, we will refer to $\min_\varphi$ and $\max_\varphi$ when evaluated by the t-norm T as $\min_T$ and $\max_T$ and will do so also for other fuzzy operators. We find using Definition 1 that for some t-norm T, $\min_T = 0$ and $\max_T = T(\boldsymbol{c})$, where $\boldsymbol{c}$ is the values of the constants $C_1, ..., C_m$ as a truth vector, while for some t-conorm S, $\min_S = S(\boldsymbol{c})$ and $\max_S = 1$. Note that for $m = 0$, $\max_T = 1$ and $\min_S = 0$. Next, we find some useful transformations of minimal refinement functions to derive new results:

Proposition 1 *Consider the formulas $\phi = \bigwedge_{i=1}^{n} P_i \bigwedge_{i=1}^{m} C_i$ and $\psi = \neg(\bigvee_{i=1}^{n} P_i \bigvee_{i=1}^{m} C_i)$. Assume ρ^*_ϕ is a minimal refinement function for f_ϕ evaluated using t-norm T. Consider $f_\psi(\boldsymbol{t})$ evaluated using dual t-conorm S of T. Then $\rho^*_\psi(\boldsymbol{t}, \hat{t}_\psi) = \mathbf{1} - \rho^*_\phi(\mathbf{1} - \boldsymbol{t}, \hat{t}_\psi)$ is a minimal refinement function for f_ψ.*

Proof First, note $f_\psi(\boldsymbol{t}) = 1 - S(\boldsymbol{t}, \boldsymbol{c}) = 1 - (1 - T(\mathbf{1} - \boldsymbol{t}, \mathbf{1} - \boldsymbol{c})) = T(\mathbf{1} - \boldsymbol{t}, \mathbf{1} - \boldsymbol{c})$. Consider $\boldsymbol{t}' = \mathbf{1} - \boldsymbol{t}$. By the assumption of the proposition, $\rho^*_\phi(\boldsymbol{t}', \hat{t}_\phi)$ is a minimal refinement function for $T(\boldsymbol{t}', \mathbf{1} - \boldsymbol{c}) = T(\mathbf{1} - \boldsymbol{t}, \mathbf{1} - \boldsymbol{c}) = f_\psi(\boldsymbol{t})$. Furthermore, note that

$$f_\psi(\rho^*_\psi(\boldsymbol{t}, \hat{t}_\psi)) = T(\mathbf{1} - \rho^*_\psi(\boldsymbol{t}, \hat{t}_\psi), \mathbf{1} - \boldsymbol{c}) = T(\rho^*_\phi(\boldsymbol{t}', \hat{t}_\psi), \mathbf{1} - \boldsymbol{c}) = \hat{t}_\psi$$

□

An analogous argument can be made for $\phi' = \bigvee_{i=1}^{n} P_i \bigvee_{i=1}^{m} C_i$ and $\psi = \neg(\bigwedge_{i=1}^{n} P_i \bigwedge_{i=1}^{m} C_i)$ to show that, given minimal refinement function $\rho^*_{\phi'}$ of dual t-conorm S, the minimal refinement function for $f_\psi(\boldsymbol{t})$ is $\rho^*_\psi(\boldsymbol{t}, \hat{t}_\psi) = \mathbf{1} - \rho^*_\phi(\mathbf{1} - \boldsymbol{t}, \hat{t}_\psi)$.

We will use this result to simplify the process of finding minimal refinement functions for the t-norms and dual t-conorms. For example, assume we have a minimal refinement function ρ^*_T for $\hat{t}_T \in [T(\boldsymbol{t}), \max_T]$. Let S be the corresponding dual t-conorm. Then, we can change the constraint $S(\hat{\boldsymbol{t}}, \boldsymbol{c}) = \hat{t}_S$ in Eq. 7 to the equivalent constraint $\mathbf{1} - S(\hat{\boldsymbol{t}}, \boldsymbol{c}) = \mathbf{1} - \hat{t}_S$. We then use Proposition 1 to find the minimal refined vector for $\hat{t}_S \in [\min_S, S(\boldsymbol{t})]$ as $\mathbf{1} - \rho^*_T(\mathbf{1} - \boldsymbol{t}, 1 - \hat{t}_S)$.

Proposition 2 *Consider the formulas $\phi = P_1 \vee P_2$ and $\psi = \neg P_1 \vee P_2$. Assume ρ^*_ϕ is a minimal refinement function for f_ϕ evaluated using the t-conorm S, and define $\boldsymbol{t}' = [1 - t_1, t_2]$. Then $\rho^*_\psi(\boldsymbol{t}, \hat{t}_\psi) = \left[1 - \rho^*_\phi(\boldsymbol{t}', \hat{t}_\psi)_1, \rho^*_\phi(\boldsymbol{t}', \hat{t}_\psi)_2\right]^\top$ is a minimal refinement function for f_ψ.*

Proof First, note $f_\psi(\boldsymbol{t}) = S(1 - \boldsymbol{t}_1, \boldsymbol{t}_2)$. By the assumption of the proposition, $\rho^*_\phi(\boldsymbol{t}', \hat{t}_\psi)$ is a minimal refinement function for $S(\boldsymbol{t}') = f_\psi(\boldsymbol{t})$. Furthermore, note that

$$\begin{aligned} f_\psi(\rho^*_\psi(\boldsymbol{t}, \hat{t}_\psi)) &= S(1 - \rho^*_\psi(\boldsymbol{t}', \hat{t}_\psi)_1, \rho^*_\psi(\boldsymbol{t}', \hat{t}_\psi)_2) \\ &= S(1 - (1 - \rho^*_\phi(\boldsymbol{t}', \hat{t}_\psi)_1), \rho^*_\phi(\boldsymbol{t}', \hat{t}_\psi)_2) = S(\rho^*_\phi(\boldsymbol{t}', \hat{t}_\psi)) = \hat{t}_\psi. \end{aligned}$$

□

Similar to the previous proposition, this proposition gives us a simple procedure for finding the minimal refinement functions for the S-implication of some t-conorm.

7.2 Basic T-norms

In this section, we introduce the minimal refinement functions for the t-norms and t-conorms of the three main fuzzy logics (Gödel, Łukasiewicz, and Product). In particular, we consider when these t-norms and t-conorms can act on both propositions and constants, that is, $\varphi = \bigwedge_{i=1}^{n} t_i \bigwedge_{i=1}^{m} C_i$, which is evaluated with $T(\boldsymbol{t}, \boldsymbol{c})$. We present the main results with simple examples.

7.2.1 Gödel t-norm

In this section, we derive minimal refinement functions for the Gödel t-norm and t-conorm for the family of p-norms.

Proposition 3 *The minimal refinement function of the Gödel t-norm for* $\hat{t}_{T_G} \in [0, \min_{i=1}^{m} C_i]$ *is*

$$\rho^*_{T_G}(\boldsymbol{t}, \hat{t}_{T_G})_i = \begin{cases} \hat{t}_{T_G} & \text{if } \hat{t}_{T_G} \geq T_G(\boldsymbol{t}) \text{ and } t_i < \hat{t}_{T_G}, \\ \hat{t}_{T_G} & \text{if } \hat{t}_{T_G} < T_G(\boldsymbol{t}) \text{ and } i = \arg\min_{j=1}^{n} t_j, \\ t_i & \text{otherwise,} \end{cases} \tag{8}$$

The minimal refinement function of the Gödel t-conorm and $\hat{t}_{S_G} \in [\max_{i=1}^{m} C_i, 1]$ *is*

$$\rho^*_{S_G}(\boldsymbol{t}, \hat{t}_{S_G})_i = \begin{cases} \hat{t}_{S_G} & \text{if } \hat{t}_{S_G} \geq S_G(\boldsymbol{t}) \text{ and } i = \arg\max_{j=1}^{m} t_j, \\ \hat{t}_{S_G} & \text{if } \hat{t}_{S_G} < S_G(\boldsymbol{t}) \text{ and } t_i > \hat{t}_{S_G}, \\ 0 & \text{otherwise.} \end{cases} \tag{9}$$

Proof Follows from Propositions 1, 10 and 11, see Appendix A.1.1 and 1. □

Proposition 4 *A minimal refinement function of the Gödel implication* $R_G(t_1, t_2) = \begin{cases} t_2 & \text{if } t_1 > t_2, \\ 1 & \text{otherwise.} \end{cases}$ *for* $\hat{t}_{R_G} \in [\min_{R_G}, \max_{R_G}]$ *is*

$$\rho^*_{R_G}(t_1, t_2, \hat{t}_{R_G}) = \begin{cases} [\max(\hat{t}_{R_G} + \epsilon, t_1), \hat{t}_{R_G}]^\top & \text{if } \hat{t}_{R_G} < 1 \\ [t_1, \max(t_1, t_2)]^\top & \text{otherwise.} \end{cases} \tag{10}$$

where ϵ *is an arbitrarily small positive number.*

The proof is in Appendix A.1.3.

The bar plot in Fig. 4a shows an example for the Gödel t-conorm with four literals. The minimal refined vector is represented with the orange boxes, while the initial and refinement values of the entire formula are represented as a blue and purple line respectively. Here, our goal is to increase the value of the t-conorm, i.e., the maximum value. Increasing other literals up to $\hat{t}_\varphi$ would require longer orange bars and bigger values for the L_p norm. Figure 4b represents when multiple literals have the largest truth value. Here, only one

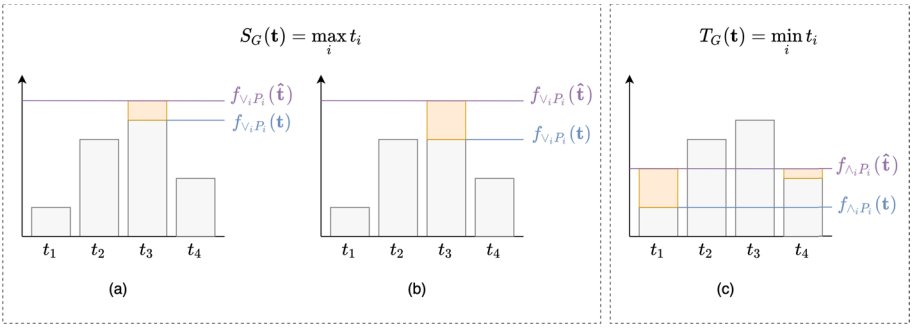

Fig. 4 Gödel minimal refinement functions. The grey bars represent the initial truth vectors $\boldsymbol{t}$; the light blue and purple lines indicate the initial truth value of the formula and the revision value $\hat{t}_\varphi$, and the orange bars are the corresponding minimal refined vectors. **a** t-conorm; **b** t-conorm with two literals with same truth value; **c** t-norm (Color figure online)

should be increased[3]. Finally, Fig. 4c shows the refined vector for the Gödel t-norm. Since the smallest truth value should be at least $\hat{t}_\varphi$, we simply ensure all truth values are at least $\hat{t}_\varphi$.

Our results are closely related to that of Giunchiglia and Lukasiewicz (2021), which considers hard constraints, i.e., $\hat{t}_\varphi = 1$. In the hierarchical multi-label classification setting, the authors introduce an output layer that ensures predictions satisfy a set of hierarchy constraints. This layer corresponds to applications of the minimal refinement function for the Gödel implication with $\hat{t}_{R_G} = 1$. Furthermore, (Giunchiglia & Lukasiewicz, 2021) introduces CCN(h). This method considers an output layer that ensures predictions satisfy background knowledge expressed in a stratified normal logic program. The authors introduce an iterative algorithm that computes the minimal solution for such programs. This algorithm is related to that of ILR in Sect. 5. However, their formalization differs somewhat from ours, and future work could study whether these results also hold for our formalization of minimal refinement functions and if they can be extended to any value of $\hat{t}_\varphi$. Finally, (Giunchiglia & Lukasiewicz, 2021) introduces a loss function compensating for gradient bias introduced by the constrained output layer.

7.2.2 Łukasiewicz t-norm

In this section, we derive minimal refinement functions for the Łukasiewicz t-norm and t-conorm, for the family of p-norms. We will start using the following notation here: $\boldsymbol{t}^\uparrow$ refers to the truth values t_i sorted in ascending order, while $\boldsymbol{t}^\downarrow$ refers to the truth values sorted in descending order.

Proposition 5 *Let* $\hat{t}_{T_L} \in [0, \max(\|\boldsymbol{c}\|_1 - (m-1), 0)]$ *and define* $\lambda_K = \frac{\hat{t}_{T_L} + m + K - 1 - \|\boldsymbol{c}\|_1 - \sum_{i=1}^{K} t_i^\uparrow}{K}$. *Let* K^* *be the largest integer* $1 \leq K \leq n$ *such that* $\lambda_K < 1 - t_K^\uparrow$. *Then the minimal refinement vector of the Łukasiewicz t-norm is*

[3] In our experiments, we choose randomly.

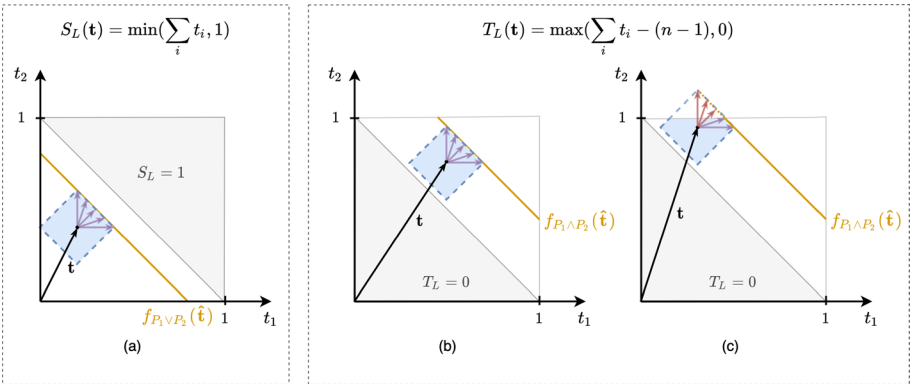

Fig. 5 Łukasiewicz minimal refinement functions. The orange line corresponds to the contour line of the S_L and T_L at the value $\hat{t}_\varphi$. The dotted blue circumference corresponds to a set of points at an equal distance from $\boldsymbol{t}$. **a** t-conorm; **b** t-norm; **c** t-norm in the limit case

$$\rho^*_{T_L}(\boldsymbol{t}, \hat{t}_{T_L})_i = \begin{cases} t_i + \lambda_{K^*} & \text{if } \hat{t}_{T_L} > T_L(\boldsymbol{t}) \text{ and } t_i \leq t^{\uparrow}_{K^*}, \\ 1 & \text{if } \hat{t}_{T_L} > T_L(\boldsymbol{t}) \text{ and } t_i > t^{\uparrow}_{K^*}, \\ t_i - \frac{\max(\|\boldsymbol{t}\|_1 + \|\boldsymbol{c}\|_1 + 1 - n - \hat{t}_{T_L}, 0)}{n} & \text{otherwise.} \end{cases} \tag{11}$$

Let $\hat{t}_{S_L} \in [\min(\|\boldsymbol{c}\|_1, 1), 1]$ *and define* $\lambda_K = \frac{\|\boldsymbol{t}\|_1 + \|\boldsymbol{c}\|_1 - \hat{t}_{S_L}}{K}$. *Let* K^* *be the largest integer* $1 \leq K \leq n$ *such that* $\lambda_K < t^{\downarrow}_K$. *Then the minimal refinement function of the Łukasiewicz t-conorm is*

$$\rho^*_{S_L}(\boldsymbol{t}, \hat{t}_{S_L})_i = \begin{cases} t_i + \frac{\max(\hat{t}_{S_L} - \|\boldsymbol{t}\|_1 - \|\boldsymbol{c}\|_1, 0)}{n} & \text{if } \hat{t}_{S_L} > S_L(\boldsymbol{t}), \\ t_i - \lambda_{K^*} & \text{if } \hat{t}_{S_L} < S_L(\boldsymbol{t}) \text{ and } t_i \geq t^{\downarrow}_{K^*}, \\ 0 & \text{otherwise.} \end{cases} \tag{12}$$

Proof This follows from Propositions 1, 12 and 13, see Appendix A.2.1 and A.2.2. □

Although slightly obfuscated, these refinement functions simply increase each of the literals equally, while properly dealing with constraints on the truth values. We explain this using Fig. 5, where the optimal solution corresponds to a vector that, from the original truth values $\boldsymbol{t}$, is perpendicular to the contour line of the operator at the value $\hat{t}_\varphi$. Moreover, the figure also provides some intuition for our proofs. The stationary points of the Lagrangian correspond to the points where the constraint function (blue circumference) tangentially touches the contour line of the refined value (orange line).

The change applied by the refinement function is proportional to the refinement value $\hat{t}$. Computing these refinement functions requires finding K^*, which can be done efficiently in log-linear time using a sort on the input truth values and a binary search.

The residuum of the Łukasiewicz t-norm is equal to its S-implication formed using $S_L(1-a, c)$, and so its minimal refinement function can be found using Proposition 2.

The Łukasiewicz logic is unique in containing large convex and concave fragments (Giannini et al., 2019). In particular, any CNF formula interpreted using the weak conjunction (Godel t-norm) and Łukasiewicz t-conorm is concave, allowing for efficient

maximization using a quadratic program of a slightly relaxed variant of the problem in Eq. 7. (Giannini et al., 2019) studies this property in a setting similar to ours in the context of collective classification. Future work could study using this convex fragment to find minimal refinement functions for more complex formulas.

7.2.3 Product t-norm

To present the three basic t-norms together, we give the closed-form refinement function for the product t-norm with the L_1 norm. Our proof is a special case of the general results on a large class of t-norms we will discuss in Sect. 7. In particular, the product t-norm is a strict, Schur-concave t-norm with an additive generator. It is an example of a t-norm for which we can find a closed-form refinement function for the L_1 norm using Propositions 15 and 1. First, we show the minimal refinement function for the product t-norm.

$$\rho^*_{T_P}(\boldsymbol{t}, \hat{t}_{T_P})_i = \begin{cases} \sqrt[n-K^*]{\frac{\hat{t}_{T_P}}{\prod_{j=1}^{K^*} t_j^{\downarrow} \prod_{j=1}^{m} C_j}} & \text{if } T_P(\boldsymbol{t}, \boldsymbol{c}) > \hat{t}_{T_P} \text{ and } t_i \leq t^{\downarrow}_{K^*+1}, \\ \frac{\hat{t}_{T_P}}{\prod_{j\neq i} t_i^{\downarrow} \prod_{i=1}^{m} C_i} & \text{if } T_P(\boldsymbol{t}, \boldsymbol{c}) < \hat{t}_{T_P} \text{ and } i = \arg\min_{j=1}^{n} t_j, \\ t_i & \text{otherwise.} \end{cases} \tag{13}$$

Next, we present the result for the product t-conorm:

$$\rho^*_{S_P}(\boldsymbol{t}, \hat{t}_{S_P})_i = \begin{cases} 1 - \frac{1-\hat{t}_{S_P}}{\prod_{j\neq i} 1-t_i^{\downarrow} \prod_{i=1}^{m} 1-C_i} & \text{if } S_P(\boldsymbol{t}, \boldsymbol{c}) < \hat{t}_{S_P} \text{ and } i = \arg\min_{j=1}^{n} t_j, \\ 1 - \sqrt[n-K^*]{\frac{1-\hat{t}_{S_P}}{\prod_{j=1}^{K^*} 1-t_j^{\downarrow} \prod_{j=1}^{m} 1-C_j}} & \text{if } S_P(\boldsymbol{t}, \boldsymbol{c}) > \hat{t}_{S_P} \text{ and } t_i \leq t^{\downarrow}_{K^*+1}, \\ t_i & \text{otherwise.} \end{cases} \tag{14}$$

This refined function increases all the literals smaller than a certain threshold up to the threshold itself, where we assume $\hat{t}_{T_P}$ is greater than the initial truth value. In fact, like the other t-norms in the class discussed in Sect. 8, it is similar to the Gödel t-norm in that it increases all literals above some threshold to the same value. Similarly, the refinement function for the t-conorm increases the highest literal. Figure 6 gives an intuition behind this behavior.

Finally, the residuum minimal refinement function can be found with $\rho^*_{I_P}(t_1, t_2, \hat{t}_{I_P}) = [t_1, \frac{\hat{t}_{I_P}}{t_1}]^\top$.

We also studied the minimal refinement function for the L_2-norm, but concluded that the result is a $2n$th degree polynomial with no simple closed-form solutions. For details, see Appendix D

8 A general class of t-norms with analytical minimal refinement functions

In this section, we will introduce and discuss a general class of t-norms that have analytic solutions to the problem in Eq. 7 to find their corresponding minimal refinement functions. We can find those for the t-norm, the t-conorm, and the residuum.

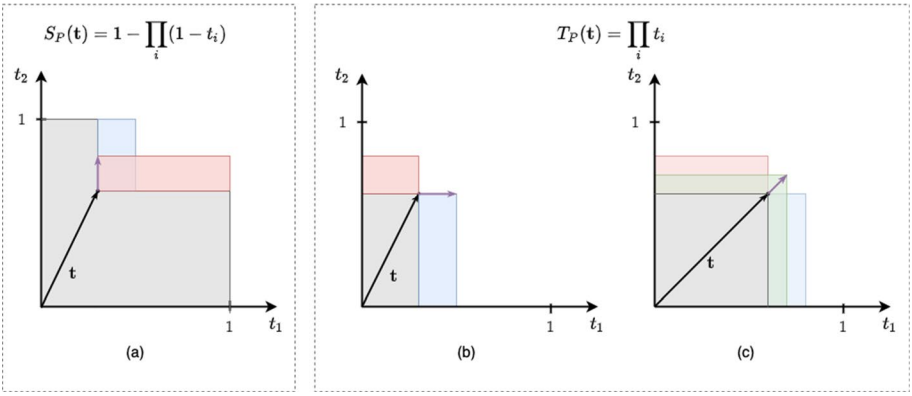

Fig. 6 Product minimal refinement functions. The grey areas represent the truth value of the operator associated with the initial vector $\boldsymbol{t}$. Red and blue areas represent the refined values when increasing a single literal. **a** t-conorm; **b** t-norm; **c** t-norm when multiple literals have the same truth value. The green area represents the improvement obtained by increasing both literals equally (Color figure online)

8.1 Background on t-norms

To be able to adequately discuss this class of t-norms, we first have to provide some more background on the theory of t-norms.

Definition 6 A t-norm T is *Archimedean* if for all $x, y \in (0, 1)$, there is an n such that $\underbrace{T(x, \ldots, x)}_{n\times} < y$.

A continuous t-norm T is *strict* if, in addition, for all $x \in (0, 1)$, $0 < T(x, x) < x$.

8.1.1 Additive generators

The study of t-norms frequently involves the study of their *additive generator* (Klement et al., 2000, 2004), which are univariate functions that construct t-norms, t-conorms, and residuums.

Definition 7 A function $g : [0, 1] \to [0, \infty]$ such that $g(1) = 0$ is an *additive generator* if it is strictly decreasing, right-continuous at 0, and if for all $t_1, t_2 \in [0, 1]$, $g(t_1) + g(t_2)$ is either in the range of g or in $[g(0^+), \infty]$.

Theorem 1 *If g is an additive generator, then the function $T : [0, 1]^n \to [0, 1]$ defined as*

$$T(\boldsymbol{t}) = g^{-1}(\min(g(0^+), \sum_{i=1}^{n} g(t_i))) \tag{15}$$

is a t-norm.

Using Eq. 15, the function g acts like an invertible function. It transforms truth values into a new space that can be seen as measuring 'untruthfulness'. $\sum_{i=1}^{n} g(t_i)$ can be seen as

a measure of the 'untruth' of the conjunction. T-norms constructed in this way are necessarily Archimedean, and each continuous Archimedean t-norm has an additive generator. T_P, T_L and T_D have an additive generator, but T_G and T_N do not. Furthermore, if $g(0^+) = \infty$, T is strict and we find $T(\boldsymbol{t}) = g^{-1}(\sum_{i=1}^{n} g(t_i))$. The residuum constructed from continuous t-norms with an additive generator can be computed using $g^{-1}(\max(g(c) - g(a), 0))$ (Jayaram & Baczynski, 2008).

8.1.2 Schur-concave t-norms

We will frequently consider the class of Schur-concave t-norms, with their dual t-conorms and residuums formed from these Schur-concave t-norms. We denote with $\boldsymbol{t}^{\downarrow}$ the truth vector $\boldsymbol{t}$ sorted in descending order, and with $\boldsymbol{t}^{\uparrow}$ as $\boldsymbol{t}$ sorted in ascending order.

Definition 8 A vector $\boldsymbol{t} \in \mathbb{R}^n$ is said to *majorize* another vector $\boldsymbol{u} \in \mathbb{R}^n$, denoted $\boldsymbol{t} \succ \boldsymbol{u}$, if $\sum_{i=1}^{n} t_i = \sum_{i=1}^{n} u_i$ and if for each $i \in \{1, ..., n\}$ it holds that $\sum_{j=1}^{i} t_j^{\downarrow} \geq \sum_{j=1}^{i} u_j^{\downarrow}$.

Definition 9 A function $[0, 1]^n \to [0, 1]$ is called *Schur-convex* if for all $\boldsymbol{t}, \boldsymbol{u} \in [0, 1]^n$, $\boldsymbol{t} \succ \boldsymbol{u}$ implies that $f(\boldsymbol{t}) \geq f(\boldsymbol{u})$. Similarly, a *Schur-concave* function has that $\boldsymbol{t} \succ \boldsymbol{u}$ implies that $f(\boldsymbol{t}) \leq f(\boldsymbol{u})$.

The dual t-conorm of a Schur-concave t-norm is Schur-convex. The three basic and continuous t-norms T_G, T_P and T_L are Schur-concave. There are also non-continuous Schur-concave t-norms, such as the Nilpotent minimum (Takači, 2005; van Krieken et al., 2022). The drastic t-norm is an example of a t-norm that is not Schur-concave (Takači, 2005). This class includes all quasiconcave t-norms since all symmetric quasiconcave functions are also Schur-concave (Schur-concave 2011, see p98, Prop.C.3). Therefore, this class constitutes a significant class of relevant t-norms. For a more precise characterization of Schur-concave t-norms, see (Takači, 2005; Alsina, 1984).

8.2 Minimal refinement functions for Schur-concave t-norms

We now have the background to discuss several useful and interesting results on Schur-concave t-norms. First, we present two results that characterize Schur-concave minimal refinement functions. We use the notion of "strictly cone-increasing" functions here that is discussed in Appendix B.1.

Theorem 2 *Let T be a Schur-concave t-norm that is strictly cone-increasing at $\hat{t}_T$ and let $\|\cdot\|$ be a strict norm. Then there is a minimal refined vector $\boldsymbol{t}^*$ for $\boldsymbol{t}$ and $\hat{t}_T$ such that whenever $t_i > t_j$, then ${t^*}_i - t_i \leq {t^*}_j - t_j$.*

For proof, see Appendix C.1. We note that we can make this argument in the other direction to show that any Schur-convex t-conorm will have a minimal refined vector such that $t_i > t_j$ implies ${t^*}_i \geq {t^*}_j$. Furthermore, if we know that a t-norm has a unique minimal refinement function, we can use this theorem to infer a useful ordering on how it changes the truth values.

Next, we will consider the L_1 norm $\sum_{i=1}^{n} |\hat{t}_i - t_i|$, for which we can find general solutions for the t-norm, t-conorm and R-implication when the t-norm is Schur-concave.

Proposition 6 *Let $\boldsymbol{t} \in [0,1]^n$ and let T be a Schur-concave t-norm that is strictly cone-increasing at $\hat{t}_T \in [T(\boldsymbol{t},\boldsymbol{c}), \max_T]$. Then there is a value $\lambda \in [0,1]$ such that the vector $\boldsymbol{t}^*$,*

$$t^*{}_i = \begin{cases} \lambda, & \text{if } t_i < \lambda, \\ t_i, & \text{otherwise,} \end{cases} \tag{16}$$

is a minimal refined vector for T and the L_1 norm at $\boldsymbol{t}$ and $\hat{t}_T$.

For proof, see Appendix C.2. We found this result rather surprising: It is optimal for a large class of t-norms and the L_1 norm to increase the lower truth values to some value λ. In this sense, these solutions are very similar to that of the Gödel refinement functions. The value of λ depends on the choice of t-norm and $T(\boldsymbol{t}^*,\boldsymbol{c})$ is a non-decreasing function of λ. We show in Sect. 8.3 how to compute these.

We have a similar result, proof in the end of Appendix C.2, for the refinement functions of Schur-convex t-conorms. This proposition shows that, under the L_1 norm, it is optimal to increase only the largest literal, just like with the Gödel t-norm.

Proposition 7 *Let $\boldsymbol{t} \in [0,1]^n$ and let S be a Schur-convex t-conorm that is strictly cone-increasing at $\hat{t}_S \in [S(\boldsymbol{t},\boldsymbol{c}), 1]$. Then there is a value $\lambda \in [0,1]$ such that the vector $\boldsymbol{t}^*$,*

$$t^*{}_i = \begin{cases} \lambda & \text{if } i = \arg\max_{i \in D} t_i, \\ t_i, & \text{otherwise,} \end{cases} \tag{17}$$

is a minimal refined vector for S and the L_1 norm at $\boldsymbol{t}$ and $\hat{t}_S$.

8.3 Closed forms using additive generators

Where the previous section gives general results on the form or "shape" of minimal refinement functions for t-norms and t-conorms under the L_1 norm, we still need to figure out what the value of λ is for a particular $\hat{t}_\varphi$. Luckily, additive generators will do the job here.

Proposition 8 *Let T be a Schur-concave t-norm with additive generator g and let $0 < \hat{t}_T \in [T(\boldsymbol{t},\boldsymbol{c}), \max_T]$. Let $K \in \{0,...,n-1\}$ denote the number of truth values such that $t^*{}_i = t_i$ in Eq. 28.*

Then using

$$\lambda_K = g^{-1}\left(\frac{1}{n-K}\left(g(\hat{t}_T) - \sum_{i=1}^{K} g(t_i^{\downarrow}) - \sum_{i=1}^{m} g(C_i)\right)\right) \tag{18}$$

in Eq. 28 gives $T(\boldsymbol{t}^*,\boldsymbol{c}) = \hat{t}_T$ if $\boldsymbol{t}^* \in [0,1]^n$.

See Appendix C.2 for a proof. $g(\hat{t}_T)$ can be seen as the 'untruth'-value in g-space that $\boldsymbol{t}^*$ should attain. Since we have $n-K$ truth values that we can move freely, we need to make sure that their 'untruth'-value in g-space is $g(\hat{t}_T)/(n-K)$. However, we also need to handle the truth values we cannot change freely, which is why those are subtracted from $g(\hat{t}_T)$.

We should note that this does not yet give a procedure for computing the correct $K \in \{0,...,n-1\}$. The intuition here is that we should find an K such that $t_i \geq \lambda_K$ for the

K largest values, and $t_i < \lambda_K$ for the remaining $n - K$. Like with computing the K^* for the refinement function for the Łukasiewicz t-norm (Sect. 7.2.2), we can do this in logarithmic time after sorting $\boldsymbol{t}$, but we choose to compute λ_K for each $K \in \{0, n-1\}$ in parallel.

We can similarly find a closed form for the t-conorms:

$$\lambda = 1 - g^{-1}\left(g(1-\hat{t}_S) - \sum_{i \neq j} g(1-t_i) - \sum_{i=1}^{m} g(1-C_i) \right) \tag{19}$$

Proposition 9 *Let $t_1, t_2 \in [0,1]$ and let T be a strict Schur-concave t-norm with additive generator g. Consider its residuum $R(t_1,t_2) = \sup\{z | T(t_1,z) \leq t_2\}$ that is strictly cone-increasing at $0 < \hat{t}_R \in [R(t_1,t_2), \max_R]$. Then there is a value $\lambda \in [0,1]$ such that $\boldsymbol{t}^* = [t_1, g^{-1}(g(\hat{t}_R) + g(t_1))]^\top$ is a minimal refined vector for R and the L_1 norm at $\boldsymbol{t}$ and t.*

Here, we find that for this class of residuums, increasing the consequent (the second argument of the implication) is minimal for the L_1 norm. This update reflects modus ponens reasoning: When the antecedent is true, increase the consequent. As we have argued in van Krieken et al. (2022), this could cause issues in many machine learning setups: Consider the modus tollens correction instead decreases the antecedent. For common-sense knowledge, this is more likely to reflect the true state of the world.

9 Experiments

We performed experiments on two tasks. The first one does not involve learning. Instead, we aim to solve SAT problems. This experiment allows assessing whether ILR can enforce complex and unstructured knowledge. The second experiment is on the MNIST Addition task (Manhaeve et al., 2018) to test ILR in a neurosymbolic setting and assess its ability to learn from data.

9.1 Experiments on 3SAT problems

With this experiment, we aim to determine how quickly ILR finds a refined vector and how minimal this vector is. We test this on formulas of varying complexity to analyze for what problems each algorithm performs well.[4]

9.1.1 Setup

We perform experiments on SATLIB (Hoos, 2000), a library of randomly generated 3SAT problems. 3SAT problems are formulas in the form $\bigwedge_{i=1}^{c} \bigvee_{j=1}^{3} l_{ij}$, where l_{ij} is a literal that is either P_k or $\neg P_k$ and where $P_k \in \{P_1, ..., P_n\}$ is an input proposition. In particular, we consider uf20-91 of satisfiable 3SAT problems with $n = 20$ propositions and $c = 91$ disjunctive clauses. For this, we select the refined value $\hat{t}_\varphi$ to be 1. We also experiment with $\hat{t}_\varphi \in \{0.3, 0.5\}$ in Appendix E . We uniformly generate initial truth values for the

[4] Code available at https://github.com/DanieleAlessandro/IterativeLocalRefinement.

propositions $\boldsymbol{t} \in [0,1]^d$.[5] To allow experimenting with formulas of varying complexity, we introduce a simplified version of the task which uses only the first 20 clauses.

We use three metrics to compare ILR with a gradient descent baseline described in Sect. 9.1.2. The first is speed: How many iterations does it take for each algorithm to converge? Since both algorithms have similar computational complexities, we will use the number of iterations for this. The second is satisfaction: Is the algorithm able to find a solution with truth value $\hat{t}_\varphi$? Finally, we consider minimality: How close to the original prediction is the refined vector $\hat{\boldsymbol{t}}$? Note that the refinement function for the product logic is only optimal for the L_1 norm, while for Gödel and Łukasiewicz, the refinement function is optimal for all L_p norms, including L_1. Moreover, the results of L_1 and L_2 are very similar. Therefore, we use the L_1 as a metric for minimality for each t-norm.

9.1.2 Gradient descent baseline

We compare ILR to gradient descent with the following loss function

$$\mathcal{L}(\hat{\mathbf{z}}, \boldsymbol{t}, \hat{t}_\varphi) = \|f_\varphi(\sigma(\hat{\mathbf{z}})) - \hat{t}_\varphi\|_2 + \beta \|\sigma(\hat{\mathbf{z}}) - \boldsymbol{t}\|_p. \tag{20}$$

Here $\hat{\boldsymbol{t}} = \sigma(\hat{\mathbf{z}})$ is a real-valued vector $\hat{\mathbf{z}} \in \mathbb{R}^n$ transformed to $\hat{\boldsymbol{t}} \in [0,1]^n$ using the sigmoid function σ to ensure the values of $\hat{\boldsymbol{t}}$ remain in $[0,1]^n$ during gradient descent. The first term minimizes the distance between the current truth value of the formula φ and the refinement value. In contrast, the second term is a regularization term that minimizes the distance between the refined vector and the original truth value $\boldsymbol{t}$ in the L_p norm. β is a hyperparameter that trades off the importance of this regularization term.

This method for finding refined vectors is very similar to the collective classification method introduced in SBR (Diligenti et al., 2017; Roychowdhury et al., 2021). The main difference is in the L_p norms chosen, as we use squared error for the first term instead of the L_1 norm. Gradient descent is a steepest descent method that takes steps minimizing the L_2 norm. Therefore, it can also be seen as a method for finding minimal refinement functions given the L_2 norm. The coordinate descent algorithm is the corresponding steepest descent method for the L_1 norm. Future work could compare how coordinate descent performs for finding minimal refinement functions for the L_1 norm. We suspect it will be much slower than gradient descent-based methods as it can only change a single truth value each iteration.

We found that ADAM (Kingma & Ba, 2015) significantly outperformed standard gradient descent in all metrics, and we chose to use it throughout our experiments. Furthermore, inspired by the analysis of the derivatives of aggregation operators in van Krieken et al. (2022), we slightly change the formulation of the loss function for the Łukasiewicz t-norm and product t-norm. The Łukasiewicz t-norm will have precisely zero gradients for most of its domain. Therefore, we remove the max operator when evaluating the $\bigwedge$ in the SAT formula, so it has nonzero gradients. For the product t-norm, the gradient will also approach 0 because of the large set of numbers between [0, 1] that it multiplies. As suggested by van Krieken et al. (2022), we instead optimize the logarithm of the product t-norm:

[5] Each run used the same initial value for each algorithm to have a fair comparison.

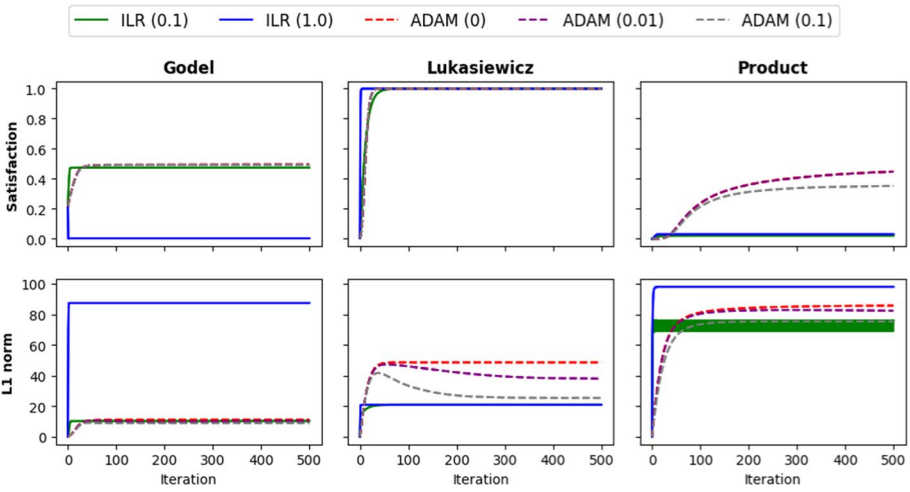

Fig. 7 Comparison of ILR with ADAM on uf20-91 in SATLIB. The target Refined value is 1.0. The x-axis corresponds to the number of iterations, while the y-axis is the value of $\hat{t}_\varphi$ in the first row of the grid and the L_1 norm in the second row. The number in parentheses represents the schedule parameter for ILR and the regularization parameter β for ADAM. Note that ADAM's plots often almost perfectly overlap

$$\mathcal{L}_P(\hat{\mathbf{z}}, \boldsymbol{t}, \hat{t}_\varphi) = \| \sum_{i=1}^{c} \log f_{\bigvee_{j=1}^{3}}(\sigma(\boldsymbol{t})) - \log \hat{t}_\varphi \|_2 + \beta \| \sigma(\hat{\mathbf{z}}) - \boldsymbol{t} \|_1 .$$

9.1.3 Results

In Fig. 7, we show the results obtained by ILR and ADAM on the three t-norms (one for each grid column). We observe that ILR with schedule parameter $\alpha = 0.1$ has a smoother plot than ILR with $\alpha = 1.0$, which converges faster: In our experiments, the number of steps until convergence was always between 2 and 5. For both values of the scheduling parameters, ILR outperforms ADAM in terms of convergence speed.

When comparing satisfaction and minimality, the behaviour differs based on the t-norm. In the case of Łukasiewicz, all methods find feasible solutions to the optimization problem. Furthermore, in terms of minimality (i.e., L_1 norm), ILR finds better solutions than ADAM.

For the Gödel logic, no method can reach a feasible solution. Here, ILR with schedule parameter $\alpha = 1$ performs very poorly, obtaining worse solutions than the original truth values. On the other hand, with $\alpha = 0.1$, it performs as well as ADAM for both metrics but with faster convergence.

Finally, for the product logic, ILR fails to increase the satisfaction of the formula to the refined value. However, ADAM can find much better solutions, getting the average truth value to around 0.5. Still, it is far from reaching a feasible solution. Nonetheless, we recommend using ADAM for complicated formulas in the product logic.

However, we argue that in the context of Neural-Symbolic Integration, the provided knowledge is usually relatively easy to satisfy. With 91 clauses, there are few satisfying solutions in this space of 2^{21} possible binary solutions. However, background knowledge usually does not constrain the space of possible solutions as heavily as this. For this reason,

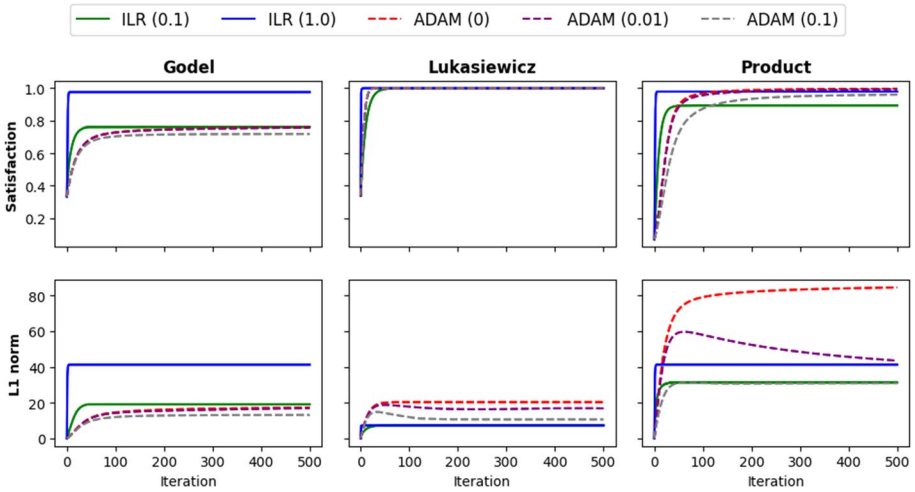

Fig. 8 Comparison of ILR with ADAM on the uf20-91 with 20 clauses. Target value 1.0

we propose a simplified formula, where we only use 20 out of 91 clauses. Figure 8 shows the results for this setting. We see that ILR with no scheduling ($\alpha = 1$) finds feasible solutions for all t-norms. ILR finds solutions for the Gödel t-norm where ADAM cannot find any, while for Łukasiewicz and product, it finds solutions in much fewer iterations and with a lower L_1 norm. Hence, we argue that for knowledge bases that are less constraining, ILR without scheduling is the best choice.

9.2 Experiments on MNIST addition

The experiments on the SATLIB benchmark show how well ILR can enforce knowledge in highly constrained settings. However, as already mentioned, in neurosymbolic AI, the background knowledge is typically much simpler. SAT benchmarks often only have a few solutions, heavily limiting what predictions the neural network can make. Moreover, previous experiments only tested ILR where initial truth vectors are random, and we did not have any neural networks or learning.

To evaluate the performance of ILR in neurosymbolic settings, we implemented the architecture of Fig. 3. Here, the task is to learn a classifier for handwritten digits while only receiving supervision on the sums of pairs of digits.

9.2.1 Setup

We follow the architecture of Fig. 3. We use the neural network proposed by Manhaeve et al. (2018), which is a network composed of two convolutional layers, followed by a MaxPool layer, followed by a fully connected layer with ReLU activation function and a fully connected layer with softmax activation. We use the Gödel t-norm and corresponding minimal refinement functions. Note that Gödel implication can only increase the consequent and can never decrease the antecedents. For this reason, ILR converges in a single step.

Table 2 Results on the MNIST addition task.

	30000	3000
DeepProblog (Manhaeve et al., 2018)	97.20 ± 0.45	92.18 ± 1.57
LTN (Badreddine et al., 2022)	96.78 ± 0.5	92.15 ± 0.75
ILR	96.67 ± 0.45	93.38 ± 1.70

We report the accuracy of predicting the sum (in %) on the test set with 30000 and 3000 samples. DeepProbLog results are taken from Badreddine et al. (2022). LTN results have been obtained by replicating the experiments of Badreddine et al. (2022)

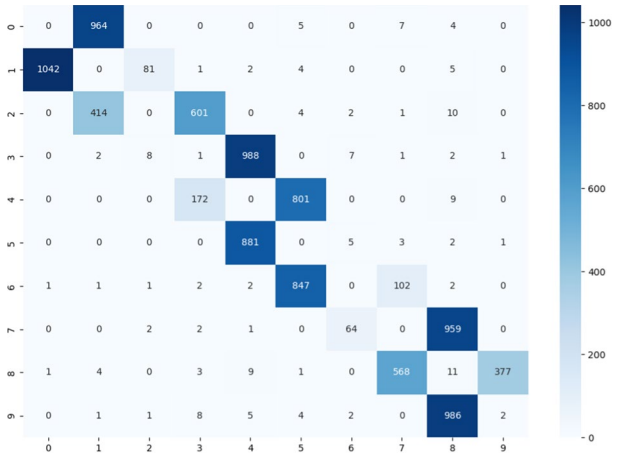

Fig. 9 Confusion matrix on the MNIST classification for a local minimum

We set both α and target value $\hat{t}$ to one, meaning that we ask ILR to satisfy the entire formula in one step. We use the ADAM optimizer and a learning rate of 0.01, with the cross-entropy loss function. However, since the outputs of the ILR step do not sum to one, we cannot directly apply it to the refined vector ILR computes. To overcome this issue, we add a logarithm followed by a softmax as the last layers of the model. If the sum of the refined vector is one, the composition of the logarithm and softmax functions corresponds to the identity function. Moreover, these two layers are monotonic increasing functions and preserve the order of the refined vector.

We use the dataset defined in Manhaeve et al. (2018) with 30000 samples, and also run the experiment using only 10% of the dataset (3000 samples). We run ILR for 5 epochs on the complete dataset, and 30 epochs on the small one. We repeat this experiment 10 times. We are interested in the accuracy obtained in the test set for the addition task. We ran the experiments on a MacBook Pro (2016) with a 3,3 GHz Dual-Core Intel Core i7.

9.2.2 Results

ILR can efficiently learn to predict the sum, reaching results similar to state of the art, requiring, on average, 30 s per epoch. However, sometimes ILR got stuck in a local minimum during training, where the accuracy reached was close to 50%. It is worth noticing that LTN suffers from

the same problem (Badreddine et al., 2022), with results strongly dependent on the initialization of the parameters. To better understand this local minimum, we analyzed the confusion matrix. Figure 9 shows one of the confusion matrices for a model stuck in the local minimum: the CNN recognizes each digit either as the correct digit minus one or plus one. Then, our model obtains the correct prediction in close to 50% of the cases. For example, suppose the digits are a 3 and a 5. The 3 is classified as a 2 or a 4, while the 5 is classified as a 4 or a 6. If the model predicts 2 and 6 or 4 and 4, it returns the correct sum (8). Otherwise, it does not. We believe that in these local minima, there is no way for the model to change the digit predictions without increasing the loss, and the model remains stuck in the local minimum.

Table 2 shows the results in terms of accuracy of ILR, LTN (Badreddine et al., 2022) and DeepProblog (Manhaeve et al., 2018). To calculate the accuracy, we follow (Badreddine et al., 2022) and select only the models that do not stop in a local minimum. Notice that this problem is rare for ILR (once every 30 runs) and happens more frequently with LTN (once every 5 runs).

10 Conclusion and future work

We analytically studied a large class of minimal fuzzy refinement functions. We used refinement functions to construct ILR, an efficient algorithm for general formulas. Another benefit of these analytical results is to get a good intuition into what kind of corrections are done by each t-norm. In our experimental evaluation of this algorithm, we found that our algorithm converges much faster and often finds better solutions than the baseline ADAM, especially for less constraining problems. However, we conclude that for complicated formulas and product logic, ADAM finds better results. Finally, we assess ILR on the MNIST Addition task and show it can be combined with a neural network, providing results similar to two of the most prominent methods for neurosymbolic AI.

There is a lot of opportunity for future work on refinement functions. We will study how the refinement functions induced by different t-norms perform in practical neurosymbolic integration settings. On the theoretical side, possible future work could be considering analytical refinement functions for certain classes of complex formulas. Furthermore, there are many classes of t-norms and norms for which finding analytical refinement functions is an open problem. Another promising avenue for research is designing specialized loss functions that handle biases in the gradients arising from combining constrained output layers with cross-entropy loss functions (Giunchiglia & Lukasiewicz, 2021). We also want to highlight the possibility of extending the work on fuzzy refinement functions to probabilistic refinement functions, using a notion of minimality such as the KL-divergence.

A Basic T-norms (Proofs)

A.1 Gödel t-norm minimal refined function proofs

A.1.1 Gödel t-norm

Proposition 10 *The minimal refinement function of the Gödel t-norm for* $\hat{t}_{T_G} \in [T_G(\boldsymbol{t}, \boldsymbol{c}), \min_{i=1}^{m} C_i]$ *is*

$$\rho^*_{T_G}(\boldsymbol{t}, \hat{t}_{T_G})_i = \begin{cases} \hat{t}_{T_G} & \text{if } t_i < \hat{t}_{T_G}, \\ t_i & \text{otherwise} \end{cases} \tag{21}$$

Proof Assume otherwise. Then there is a refined vector $\hat{\boldsymbol{t}}$ for T_G, $\boldsymbol{t} \in [0,1]^n$ and $\hat{t}_{T_G} \in [T_G(\boldsymbol{t}), \min_{i=1}^m]$ such that $\hat{\boldsymbol{t}} \neq \boldsymbol{t}^*$ while $\|\hat{\boldsymbol{t}} - \boldsymbol{t}\|_p < \|\boldsymbol{t}^* - \boldsymbol{t}\|_p$, where $\boldsymbol{t}^* = \rho^*_{T_G}(\boldsymbol{t}, \hat{t}_{T_G})$. Since $T_G(\hat{\boldsymbol{t}}) = \hat{t}_{T_G}$, for all $i \in \{1, ..., n\}$, $\hat{t}_i \geq \hat{t}_{T_G}$ and so necessarily for all i such that $t_i < \hat{t}_{T_G}$, $\hat{t}_i \geq \hat{t}_{T_G}$. Since there is some i such that $\hat{t}_i \neq {t^*}_i$, either $t_i < \hat{t}_{T_G}$ and then necessarily $\hat{t}_i > {t^*}_i$, or $\hat{t}_i \geq \hat{t}_{T_G}$ but $\hat{t}_i \neq {t^*}_i = t_i$. In either case, since $\|\cdot\|_p$ is strictly convex in each argument with minimum at $\boldsymbol{t}$, $\|\hat{\boldsymbol{t}} - \boldsymbol{t}\|_p > \|\boldsymbol{t}^* - \boldsymbol{t}\|_p$, hence $\hat{\boldsymbol{t}}$ could not have smaller norm. □

A.1.2 Gödel t-conorm

A derivation for increasing the Gödel t-conorm was first presented in Daniele and Serafini (2019) and is adapted to our notation here:

Proposition 11 *The minimal refinement function of the Gödel t-conorm for* $\hat{t}_{S_G} \in [S_G(\boldsymbol{t}, \boldsymbol{c}), 1]$ *is*

$$\rho^*_{S_G}(\boldsymbol{t}, \hat{t}_{S_G})_i = \begin{cases} \hat{t}_{S_G} & \text{if } i = \arg\max_{j=1}^n t_j \\ t_i & \text{otherwise.} \end{cases} \tag{22}$$

A.1.3 Gödel Implication

We next present a proof for Proposition 4.

Proof First, assume $\hat{t}_{R_G} < 1$. To ensure $R_G(t_1, t_2) = \hat{t}_{R_G}$, we require $t_2 = \hat{t}_{R_G}$ as is clear from the definition. However, we also require $t_1 > \hat{t}_{R_G}$. If t_1 is already larger, we can leave it to ensure minimality. Otherwise, we require it to be at least infinitesimally bigger, that is $\hat{t}_{R_G} + \epsilon$.

Next, assume $\hat{t}_{R_G} = 1$. If $t_1 \leq t_2$, then the implication is already 1 and we do not need to revise anything. Otherwise, setting it equal to any value between t_2 and t_1 is minimal. □

A.2 Łukasiewicz t-norm minimal refined function proofs

A.2.1 Łukasiewicz t-norm

Proposition 12 *Let* $\hat{t}_{T_L} \in [T_L(\boldsymbol{t}, \boldsymbol{c}), \max(\|\boldsymbol{c}\|_1 - (m-1), 0)]$ *and define* $\lambda_K = \frac{\hat{t}_{T_L} + m + K - 1 - \|\boldsymbol{c}\|_1 - \sum_{i=1}^K t_i^\uparrow}{K}$. *Let* K^* *be the largest integer* $1 \leq K \leq |D|$ *such that* $\lambda_K < 1 - t_K^\uparrow$. *Then the minimal refinement vector of the Łukasiewicz t-norm is*

$$\rho^*_{T_L}(\boldsymbol{t}, \hat{t}_{T_L})_i = \begin{cases} t_i + \lambda_{K^*} & \text{if } t_i \leq t_K^\uparrow, \\ 1 & \text{otherwise} \end{cases} \tag{23}$$

Proof We will prove this using the KKT conditions, which are both necessary and sufficient for minimality for the Łukasiewicz t-norm since it is affine when the max constraint

is not active. We drop the p-root in the norm since it is a strictly monotonically increasing function. The Lagrangian and corresponding derivative is

$$\ell = \sum_{i=1}^{n} |\hat{t}_i - t_i|^p + \lambda(\max(\|\hat{\boldsymbol{t}}\|_1 + \|\boldsymbol{c}\|_1 - (m+n-1), 0) - \hat{t}_{T_L}) + \sum_{i=1}^{n} \gamma_i(\hat{t}_i - 1)$$

$$\frac{\partial \ell}{\partial \hat{t}_i} = p(\hat{t}_i - t_i)^{p-1} + \lambda \frac{\partial}{\partial \hat{t}_i} \max(\|\hat{\boldsymbol{t}}\|_1 + \|\boldsymbol{c}\|_1 - (m+n-1), 0) + \gamma_i = 0.$$

We note that we drop the absolute signs since T_L is strictly monotonically increasing function and $\hat{t}_{T_L} \geq T_L(\boldsymbol{t}, \boldsymbol{c})$. Assuming $\hat{t}_{T_L} > 0$, $T_L(\hat{\boldsymbol{t}}, \boldsymbol{c}) = \hat{t}_{T_L}$ can only be true if the first argument of max is chosen. Then for all $i, j \in \{1, ..., n\}$, $p(\hat{t}_i - t_i)^{p-1} + \gamma_i = p(\hat{t}_j - t_j)^{p-1} + \gamma_j$. Define I as the set of K^* smallest t_i.

- *Primal feasibility:* For all $i \in I$, $\rho^*_{T_L}(\boldsymbol{t}, \hat{t}_{T_L})_i = \lambda_{K^*} \leq 1$ by definition. For all $i \in \{1, ..., n\} \setminus I$, $\rho^*_{T_L}(\boldsymbol{t}, \hat{t}_{T_L})_i = 1 - t_i$. Furthermore,

$$\begin{aligned} T_L(\rho^*_{T_L}(\boldsymbol{t}, \hat{t}_{T_L}), \boldsymbol{c}) &= \max(\sum_{i=1}^{K^*} (t_i^{\uparrow} + \lambda_{K^*}) + \sum_{i=K^*+1}^{n} 1 + \|\boldsymbol{c}\|_1 - n - m + 1, 0) \\ &= \max(\sum_{i=1}^{K^*} t_i^{\uparrow} + K^* \lambda_{K^*} + n - K^* + \|\boldsymbol{c}\|_1 - n - m + 1, 0) \\ &= \max(\sum_{i=1}^{K^*} t_i^{\uparrow} + \hat{t}_{T_L} + m + K^* - 1 - \|\boldsymbol{c}\|_1 - \sum_{i=1}^{K^*} t_i^{\uparrow} \\ &\quad - K^* + \|\boldsymbol{c}\|_1 - m + 1, 0) = \hat{t}_{T_L} \end{aligned}$$

- *Complementary Slackness:* Clearly, for all $i \in I$, we require $\gamma_i = 0$. For all $i \in \{1, ..., n\} \setminus I$, $\rho^*_{T_L}(\boldsymbol{t}, \hat{t}_{T_L} t)_i - 1 = 1 - 1 = 0$.
- *Dual feasibility:* For all $i \in I$, $\gamma_i = 0$. For $i \in \{1, ..., n\} \setminus I$, consider some $j \in I$ and note that $p(\hat{t}_i - t_i)^{p-1} + \gamma_i = p(\hat{t}_j - t_j)^{p-1} + \gamma_j$. Filling in $\hat{\boldsymbol{t}}$, we find $\gamma_i = p\lambda_{K^*}^{p-1} - p(1 - t_i)^{p-1}$. This is nonnegative if $\lambda_{K^*} \geq 1 - t_i$. First, we show $\lambda_{K^*} \geq \lambda_{K^*+1}$. Write out their definitions, multiply by $K^*(K^* + 1)$ and remove common terms. Then,

$$\begin{aligned} \hat{t}_{T_L} + m - 1 - \|\boldsymbol{c}\|_1 - \sum_{i=1}^{K^*} t_i^{\uparrow} &\geq -K^* t_{K^*+1}^{\uparrow} \\ \hat{t}_{T_L} + m + K^* + 1 - \|\boldsymbol{c}\|_1 - \sum_{i=1}^{K^*+1} t_i^{\uparrow} &\geq (K^* + 1)(1 - t_{K^*+1}^{\uparrow}) \\ \lambda_{K^*+1} &\geq 1 - t_{K^*+1}^{\uparrow}. \end{aligned}$$

  $\lambda_{K^*+1} \geq 1 - t_{K^*+1}^{\uparrow}$ is true by the construction in the proposition. Therefore,

$$\lambda_{K^*} \geq \lambda_{K^*+1} \geq 1 - t_{K^*+1}^{\uparrow} \geq 1 - t_i,$$

  proving dual feasibility.

□

A.2.2 Łukasiewicz t-conorm

Proposition 13 *The minimal refinement function of the Łukasiewicz t-conorm for* $\hat{t}_{S_L} \in [S_L(\boldsymbol{t}, \boldsymbol{c}), \hat{t}_{S_L}]$ *is*

$$\rho^*_{S_L}(\boldsymbol{t}, \hat{t}_{S_L})_i = t_i + \frac{\max(\hat{t}_{S_L} - \|\boldsymbol{t}\|_1 - \|\boldsymbol{c}\|_1, 0)}{n} \tag{24}$$

Proof We do not add multipliers for the constraints on $\hat{t}_i$, and show critical points adhere to these constraints. The Lagrangian is

$$\ell = \sum_{i=1}^{n} (\hat{t}_i - t_i)^p + \lambda(\min(\|\hat{\boldsymbol{t}}\|_1 + \|\boldsymbol{c}\|_1, 1) - \hat{t}_{S_L}) \tag{25}$$

Note that $\max_{S_L} = 1$. Taking the derivative to $\hat{t}_i$, we find

$$\frac{\partial \ell}{\partial \hat{t}_i} = p \cdot (\hat{t}_i - t_i)^{p-1} + \lambda \frac{\partial}{\partial \hat{t}_i} \min(\min(\|\hat{\boldsymbol{t}}\|_1 + \|\boldsymbol{c}\|_1, 1) = 0$$

Assume $\hat{t}_{S_L} \neq S_L(\boldsymbol{t})$, this gives three cases for all $i \in \{1, ..., n\}$:

1. If $\|\boldsymbol{t}\|_1 + \|\boldsymbol{c}\| \geq 1$ and $\hat{t}_{S_L} = 1$, then since $\hat{t}_i \geq t_i$, $\frac{\partial}{\partial \hat{t}_i} \min(\|\hat{\boldsymbol{t}}\|_1 + \|\boldsymbol{c}\|_1, 1) = \frac{\partial}{\partial \hat{t}_i} 1 = 0$, and so $\hat{t}_i = t_i$.
2. If $\|\boldsymbol{t}\|_1 + \|\boldsymbol{c}\| \geq 1$, then $\min_{S_L} = \max_{S_L} = 1$, and again $\hat{t}_i = t_i$.
3. Otherwise, it must be that $\|\hat{\boldsymbol{t}}\|_1 + \|\boldsymbol{c}\|_1 \leq 1$ and so $\frac{\partial}{\partial \hat{t}_i} \min(\|\hat{\boldsymbol{t}}\|_1 + \|\boldsymbol{c}\|_1, 1) = \frac{\partial}{\partial \hat{t}_i} \|\hat{\boldsymbol{t}}\|_1 = 1$, and therefore $p \cdot (\hat{t}_i - t_i)^{p-1} = -\lambda$. Since the equality holds for all $i \in \{1, ..., n\}$, we find $p \cdot (\hat{t}_i - t_i)^{p-1} = p \cdot (\hat{t}_j - t_j)^{p-1}$ for all $i, j \in \{1, ..., n\}$. As we are only interested in real nonnegative solutions, we find that $\hat{t}_i - t_i = \hat{t}_j - t_j = \delta$. Since $\|\hat{\boldsymbol{t}}\|_1 + \|\boldsymbol{c}\|_1 = \|\boldsymbol{t}\|_1 + \|\boldsymbol{c}\|_1 + n\delta = \hat{t}_{S_L}$, we find

$$\delta = \frac{\hat{t}_{S_L} - \|\boldsymbol{t}\|_1 - \|\boldsymbol{c}\|_1}{n}, \quad \hat{t}_i = t_i + \delta.$$

Note that $\hat{t}_i \geq t_i$, since by assumption $\hat{t}_{S_L} \geq S_L(\boldsymbol{t}, \boldsymbol{c})$, and $\hat{t}_i \leq 1$ since by $\hat{t}_{S_L} \leq \max_{S_L} \leq 1$, $\delta = \frac{\hat{t}_{S_L} - \|\boldsymbol{t}\|_1 - \|\boldsymbol{c}\|_1}{n} \leq \frac{1 - \|\boldsymbol{t}\|_1 - \|\boldsymbol{c}\|_1}{n} \leq \frac{1 - t_i}{n} \leq 1 - t_i$, that is, the constraints of Eq. 7 are satisfied.

□

B Dual problem

This section introduces a dual problem to Eq. 7. This is used extensively in several proofs.

B.1 Strict cone monotonicity

Definition 10 A set $K \subset [0,1]^n$ is a *(convex) cone* if for every $s > 0$ and $\boldsymbol{t} \in K$ such that $s\boldsymbol{t} \in [0,1]^n$, also $s\boldsymbol{t} \in K$.

A fuzzy evaluation operator f_φ is *strictly cone-increasing* at $\boldsymbol{t} \in [0,1]^n$ if there is a non-empty cone $K(\boldsymbol{t})$ such that $\boldsymbol{t}' - \boldsymbol{t} \in K$ implies $f_\varphi(\boldsymbol{t}) < f_\varphi(\boldsymbol{t}')$.

Strict cone-monotonicity is a weak notion of monotonicity in the sense that all t-norms that are strictly increasing in each argument are strictly cone-increasing, but the reverse need not be true.

Proposition 14 *If f_φ is non-decreasing and strictly cone-increasing at $\boldsymbol{t} \in [0,1]^n$, there exist a nonempty cone $K'(\boldsymbol{t}) \subseteq K(\boldsymbol{t})$ such that $\boldsymbol{t}' - \boldsymbol{t} \in K'(\boldsymbol{t})$ implies $\boldsymbol{t}'_i \geq t_i$ for all $i \in \{0, ..., n\}$.*

Proof Assume otherwise. Consider some $\boldsymbol{t}'$ such that $s(\boldsymbol{t}' - \boldsymbol{t}) \in K(\boldsymbol{t})$ for $s > 0$. By assumption, there is some $i \in \{0, ..., n\}$ such that $\boldsymbol{t}'_i < t_i$. Consider $\hat{\boldsymbol{t}}$ equal to $\boldsymbol{t}'$ except that $\hat{\boldsymbol{t}}_i = t_i$ for such i. Since f_φ is non-decreasing in each argument, $f_\varphi(\hat{\boldsymbol{t}}) \geq f_\varphi(\boldsymbol{t}') > f_\varphi(\boldsymbol{t})$, then clearly $s(\hat{\boldsymbol{t}} - \boldsymbol{t})$ for $s > 0$ forms the cone $K'(\boldsymbol{t})$. □

B.2 Dual problem

Next, we will investigate a dual problem for the problem in Eq. 7 that will allow us to prove multiple useful theorems:

$$\begin{aligned} \text{For all} \quad & \boldsymbol{t} \in [0,1]^n, u \in [0, \infty) : \\ \max_{\hat{\boldsymbol{t}}} \quad & f_\varphi(\hat{\boldsymbol{t}}) \\ \text{such that} \quad & \|\hat{\boldsymbol{t}} - \boldsymbol{t}\| = u, \\ & 0 \leq \hat{t}_i \leq 1. \end{aligned} \tag{26}$$

That is, instead of finding the $\hat{\boldsymbol{t}}$ closest to $\boldsymbol{t}$ with refinement value $\hat{t}_\varphi$, we find the largest refined value attainable with a fixed budget u. We need to be precise when solutions of this dual problem coincide with the problem in Eq. 7. We consider strict cone-monotonicity (Van Dyke et al., 2013; Clarke et al., 1993), which is a weak notion of strict monotonicity for higher dimensions. This intuitively means that there is always some direction we can move in to increase the value of the t-norm. Since t-norms are already non-decreasing in each argument, this implies there is no point where the t-norm is "flat" in all directions. The precise definition is given in Definition 10.

Theorem 3 *A solution $\boldsymbol{t}^*$ for some f_φ, $\boldsymbol{t}$ and $u \geq 0$ of Eq. 26 is also a solution to Eq. 7 for $\boldsymbol{t}$ and $\hat{t}_\varphi = f_\varphi(\boldsymbol{t}^*) \geq f_\varphi(\boldsymbol{t})$ if f_φ is non-decreasing in all arguments and strictly cone-increasing at each $\boldsymbol{t}' \in [0,1]^n$ such that $f_\varphi(\boldsymbol{t}') = \hat{t}_\varphi$, and if $\|\cdot\|$ is strictly increasing in all arguments.*

Proof Assume otherwise, and suppose a solution $\hat{\boldsymbol{t}}$ for Eq. 7 exists such that $f_\varphi(\hat{\boldsymbol{t}}) = \hat{t}_\varphi$ while $\|\hat{\boldsymbol{t}} - \boldsymbol{t}\| < \|\boldsymbol{t}^* - \boldsymbol{t}\| = u$. Since f_φ is non-decreasing in all arguments and $\hat{t}_\varphi \geq f_\varphi(\boldsymbol{t})$, $\hat{\boldsymbol{t}} - \boldsymbol{t}$ and $\boldsymbol{t}^* - \boldsymbol{t}$ are nonnegative. By Proposition 14 there is some cone $K(\hat{\boldsymbol{t}})$ that contains a line segment $\boldsymbol{\epsilon}(s) = s(\boldsymbol{t}' - \hat{\boldsymbol{t}})$ such that for all $s > 0$, $f_\varphi(\hat{\boldsymbol{t}}) < f_\varphi(\hat{\boldsymbol{t}} + \boldsymbol{\epsilon}(s))$ and for all $i \in \{0, ..., n\}$, $0 \leq \boldsymbol{\epsilon}(s)_i$. Therefore, necessarily there is some i such that $0 < \boldsymbol{\epsilon}(s)_i$. Since $\|\cdot\|$

is strictly increasing on nonnegative vectors and continuous (since it is a norm), necessarily, there are some $s > 0$ such that $\|\hat{\boldsymbol{t}} + \boldsymbol{\epsilon}(s)\| = u$. However, this is in contradiction with the premise that $\boldsymbol{t}^*$ is a solution of Eq. 26, as $f_\varphi(\hat{\boldsymbol{t}} + \epsilon(s)) > f_\varphi(\boldsymbol{t}^*)$. □

Since $f_\varphi \in [0,1]^n \to [0,1]$, f_φ cannot satisfy the conditions of Theorem 3 when $\hat{t}_\varphi = 1$. For all $\hat{t}_\varphi \in [0,1)$, however, both the Gödel and product t-norms and t-conorms are strictly cone-increasing. The Łukasiewicz t-norm satisfies the conditions for $\hat{t}_\varphi \in (0,1)$, since it has flat regions for $\hat{t}_\varphi = 0$. The same reasoning can be made for the nilpotent minimum and drastic t-norms (van Krieken et al., 2022). Furthermore, all t-norms with an additive generator are strictly cone-increasing on $\hat{t}_\varphi \in (0,1)$, as are all strict t-norms.

C Schur-concave t-norms (Proofs)

C.1 Minimal refinement function for t-norms

Theorem 4 *Let T be a Schur-concave t-norm that is strictly cone-increasing at $\hat{t}_T$ and let $\|\cdot\|$ be a strict norm. Then there is a minimal refined vector $\boldsymbol{t}^*$ for $\boldsymbol{t}$ and $\hat{t}_T$ such that whenever $t_i > t_j$, then ${t^*}_i - t_i \leq {t^*}_j - t_j$.*

Proof Assume there is a minimal refined vector $\hat{\boldsymbol{t}} \neq \boldsymbol{t}^*$ which has some $\hat{t}_i - t_i > \hat{t}_j - t_j$ while $t_i > t_j$. Consider $\hat{\boldsymbol{t}}'$ equal to $\hat{\boldsymbol{t}}$ except that $\hat{t}'_i = \hat{t}_j - t_j + t_i$ and $\hat{t}'_j = \hat{t}_i - t_i + t_j$ such that by symmetry $\|\hat{\boldsymbol{t}} - \boldsymbol{t}\| = \|\hat{\boldsymbol{t}}' - \boldsymbol{t}\|$. Define $\hat{t}'_{\max} = \max(\hat{t}'_i, \hat{t}'_j)$ and $\hat{t}'_{\min} = \min(\hat{t}'_i, \hat{t}'_j)$. Clearly, $\hat{t}_i > \hat{t}'_{\max} \geq \hat{t}'_{\min} > \hat{t}_j$.

We will show $\hat{\boldsymbol{t}}$ majorizes $\hat{\boldsymbol{t}}'$ by checking the condition of Definition 8 for any $k \in \{1, ..., n\}$.

1. If $\hat{t}^{\downarrow}_k > \hat{t}_i$, then all elements are equal and $\sum_{l=1}^k \hat{t}^{\downarrow}_l = \sum_{l=1}^k \hat{t}'^{\downarrow}_l$.
2. If $\hat{t}_i \geq \hat{t}^{\downarrow}_k > \hat{t}'_{\max}$, then $\sum_{l=1}^k \hat{t}^{\downarrow}_l = \sum_{l=1}^{k-1} \hat{t}'^{\downarrow}_l + \hat{t}_i \geq \sum_{l=1}^k \hat{t}'^{\downarrow}_l$.
3. If $\hat{t}'_{\max} \geq \hat{t}^{\downarrow}_k > \hat{t}'_{\min}$, then $\sum_{l=1}^k \hat{t}^{\downarrow}_l > \sum_{l=1}^k \hat{t}'^{\downarrow}_l$, since by removing common terms we get $\hat{t}_i > \hat{t}'_{\max}$.
4. If $\hat{t}'_{\min} \geq \hat{t}^{\downarrow}_k > \hat{t}_j$, then removing all common terms in the sums, we are left with $\hat{t}_i + \hat{t}^{\downarrow}_k > \hat{t}'_{\min} + \hat{t}'_{\max}$. Note $\hat{t}'_{\min} + \hat{t}'_{\max} = \hat{t}_j + t_i - t_j + \hat{t}_i + t_j - t_i = \hat{t}_i + \hat{t}_j$. Subtracting $\hat{t}_i$ from both sides, we are left with $\hat{t}^{\downarrow}_k > \hat{t}_j$, which is true by assumption.
5. If $\hat{t}_{\min} \geq \hat{t}^{\downarrow}_k$, then removing common terms, we are left with $\hat{t}_{\max} + \hat{t}_{\min} = \hat{t}_i + \hat{t}_j$.

Therefore, $\hat{\boldsymbol{t}}$ majorizes $\hat{\boldsymbol{t}}'$, and so by Schur concavity, $T(\hat{\boldsymbol{t}}, \boldsymbol{c}) \leq T(\hat{\boldsymbol{t}}', \boldsymbol{c})$, noting that the additional truth vector $\boldsymbol{c}$ will not influence the majorization result since it is applied at both sides. By Theorem 3, either 1) $T(\hat{\boldsymbol{t}}, \boldsymbol{c}) < T(\hat{\boldsymbol{t}}', \boldsymbol{c})$, so $\hat{\boldsymbol{t}}$ could not have been minimal, leading to a contradiction, or 2) $T(\hat{\boldsymbol{t}}, \boldsymbol{c}) = T(\hat{\boldsymbol{t}}', \boldsymbol{c})$ and both $\hat{\boldsymbol{t}}$ and $\hat{\boldsymbol{t}}'$ are minimal. □

C.2 Closed-form refinement function using additive generators

Proposition 15 *Let T be a Schur-concave t-norm with additive generator g and let $0 < \hat{t}_T \in [T(\boldsymbol{t}, \boldsymbol{c}), \max_T]$. Let $K \in \{0, ..., n-1\}$ denote the number of truth values such that ${t^*}_i = t_i$ in Eq. 28.*

Then using

$$\lambda_K = g^{-1}\left(\frac{1}{n-K}\left(g(\hat{t}_T) - \sum_{i=1}^{K} g(t_i^{\downarrow}) - \sum_{i=1}^{m} g(C_i)\right)\right) \tag{27}$$

in Eq. 28 gives $T(\boldsymbol{t}^*, \boldsymbol{c}) = \hat{t}_T$ *if* $\boldsymbol{t}^* \in [0,1]^n$.

Proof Using Eqs. 15 and 28, we find that

$$T(\boldsymbol{t}^*, \boldsymbol{c}) = g^{-1}\left(\min\left(g(0^+), \sum_{i=1}^{K} g(t_i^{\downarrow}) + \sum_{i=K+1}^{n} g(\lambda_K) + \sum_{i=1}^{m} g(C_i)\right)\right) = \hat{t}_T$$

Since $\hat{t}_T > 0$, we can remove the min, since $\hat{t}_T > 0$ will require that $\sum_{i=1}^{K} g(t_i^{\downarrow}) + (n-K)g(\lambda_K) + \sum_{i=1}^{m} g(C_i) > g(0^+)$. We apply g to both sides of the equation, which is allowed since g is a bijection. Thus

$$\begin{aligned} g(\hat{t}_T) &= \sum_{i=1}^{K} g(t_i^{\downarrow}) + (n-K)g(\lambda_K) + \sum_{i=1}^{m} g(C_i) \\ g(\lambda_K) &= \frac{1}{n-K}\left(g(\hat{t}_T) - \sum_{i=1}^{K} g(t_i^{\downarrow}) - \sum_{i=1}^{m} g(C_i)\right) \\ \lambda_K &= g^{-1}\left(\frac{1}{n-K}\left(g(\hat{t}_T) - \sum_{i=1}^{K} g(t_i^{\downarrow}) - \sum_{i=1}^{m} g(C_i)\right)\right), \end{aligned}$$

where in the last step we apply g^{-1}. □

In a similar manner, we can find the λ for the t-conorm. Let $j = \arg\max_{i=1}^{n} t_i$.

$$\begin{aligned} S(\boldsymbol{t}^*) &= 1 - g^{-1}(\min(g(0^+), \sum_{i=1}^{n} g(1-t^*{}_i) + \sum_{i=1}^{m} g(1-C_i))) = \hat{t}_S \\ \min(g(0^+), g(1-\lambda) + \sum_{i\neq j} g(1-t_i) + \sum_{i=1}^{m} g(1-C_i)) &= g(1-\hat{t}_S) \end{aligned}$$

If $\hat{t}_S < 1$, or if $g(0)$ is well defined, then we can ignore the min:

$$\begin{aligned} g(1-\lambda) &= g(1-\hat{t}_S) - \sum_{i\neq j} g(1-t_i)) - \sum_{i=1}^{m} g(1-C_i) \\ \lambda &= 1 - g^{-1}\left(g(1-\hat{t}_S) - \sum_{i\neq j} g(1-t_i) - \sum_{i=1}^{m} g(1-C_i)\right) \end{aligned}$$

C.3 L_1 minimal refinement function for t-norms

Proposition 16 *Let* $\boldsymbol{t} \in [0,1]^n$ *and let* T *be a Schur-concave t-norm that is strictly cone-increasing at* $\hat{t}_T \in [T(\boldsymbol{t}, \boldsymbol{c}), \max_T]$. *Then there is a value* $\lambda \in [0,1]$ *such that the vector* $\boldsymbol{t}^*$,

$$t^*{}_i = \begin{cases} \lambda, \text{ if } t_i < \lambda, \\ t_i, \text{ otherwise,} \end{cases} \tag{28}$$

is a minimal refined vector for T and the L_1 norm at $\boldsymbol{t}$ and $\hat{t}_T$.

Proof Assume otherwise. Then, using Theorem 3, there must be a refined vector $\hat{\boldsymbol{t}}$ such that $\|\hat{\boldsymbol{t}} - \boldsymbol{t}\|_1 = \|\boldsymbol{t}^* - \boldsymbol{t}\|_1$ but $T(\hat{\boldsymbol{t}}, \boldsymbol{c}) > T(\boldsymbol{t}^*, \boldsymbol{c})$. Since $\hat{t}_T \in [T(\boldsymbol{t}, \boldsymbol{c}), \max_T]$, we can assume $\hat{t}_i \geq t_i$.

We define $\pi^*(i)$ as the permutation in descending order of $\boldsymbol{t}^*$. Furthermore, let k be the smallest j such that $t_j^{\downarrow} < \lambda$.

Since $\|\hat{\boldsymbol{t}}\|_1 = \|\boldsymbol{t}^*\|_1$, by assumption of equal L_1 norms of $\hat{\boldsymbol{t}}$ and $\boldsymbol{t}^*$, we will prove for all $i \in \{1, ..., n\}$ that $\hat{\boldsymbol{t}}$ majorizes $\boldsymbol{t}^*$.

- If $i < k$, then $\sum_{j=1}^{i} \hat{t}_j^{\downarrow} \geq \sum_{j=1}^{i} \hat{t}_{\pi^*(j)} \geq \sum_{j=1}^{i} t_{\pi^*(j)} = \sum_{j=1}^{i} t^{*\downarrow}_j$. The first inequality follows from the fact that there is no ordering of $\hat{\boldsymbol{t}}$ that will have a higher sum than in descending order.
- If $i \geq k$, then clearly $t^{*\downarrow}_i = \lambda$. Furthermore, $\sum_{j=1}^{i} t^{*\downarrow}_j = \sum_{j=1}^{k} t_j^{\downarrow} + (i-k)\lambda$. We will distinguish two cases:

  1. $\hat{t}_i^{\downarrow} \geq \lambda$. Then for all $j \in \{k, ..., i\}$, $\hat{t}_j^{\downarrow} \geq \lambda$. Furthermore, from the previous result, $\sum_{j=1}^{k-1} \hat{t}_j^{\downarrow} \geq \sum_{j=1}^{k-1} t^{*\downarrow}_j$ and so clearly $\sum_{j=1}^{i} \hat{t}_j^{\downarrow} \geq \sum_{j=1}^{i} t^{*\downarrow}_i$.
  2. $\hat{t}_i^{\downarrow} < \lambda$. Then for all $j > i$, $\hat{t}_j^{\downarrow} \leq \hat{t}_i^{\downarrow} < \lambda$, and so $\sum_{j=i+1}^{n} \hat{t}_j^{\downarrow} \leq \sum_{j=i+1}^{n} \hat{t}_i^{\downarrow} = (n-i)\hat{t}_i^{\downarrow} < (n-i)\lambda$. Using this, we note that

  $$\|\boldsymbol{t}^*\|_1 = \sum_{j=1}^{k} t_j^{\downarrow} + (n-k)\lambda = \|\hat{\boldsymbol{t}}\|_1 = \sum_{j=1}^{i} \hat{t}_j^{\downarrow} + \sum_{j=i+1}^{n} \hat{t}_j^{\downarrow} < \sum_{j=1}^{i} \hat{t}_j^{\downarrow} + (n-i)\lambda.$$

  Then, subtracting $(n-i)\lambda$ from the inequality, we find

  $$\sum_{j=1}^{i} \hat{t}_j^{\downarrow} > \sum_{j=1}^{k} t_j^{\downarrow} + (n-k)\lambda - (n-i)\lambda = \sum_{j=1}^{k} t_j^{\downarrow} + (i-k)\lambda = \sum_{j=1}^{i} t^{*\downarrow}_j$$

And so, $\hat{\boldsymbol{t}}$ majorizes $\boldsymbol{t}^*$, and by Schur concavity of T, $T(\hat{\boldsymbol{t}}, \boldsymbol{c}) \leq T(\boldsymbol{t}^*, \boldsymbol{c})$ leading to a contradiction. □

C.4 L_1 minimal refinement function for t-conorms

Proposition 17 *Let $\boldsymbol{t} \in [0,1]^n$ and let S be a Schur-convex t-conorm that is strictly cone-increasing at $\hat{t}_S \in [S(\boldsymbol{t}, \boldsymbol{c}), 1]$. Then there is a value $\lambda \in [0,1]$ such that the vector $\boldsymbol{t}^*$,*

$$t^*{}_i = \begin{cases} \lambda \text{ if } i = \arg\max_{i \in D} t_i, \\ t_i, \text{ otherwise,} \end{cases} \tag{29}$$

is a minimal refined vector for S and the L_1 norm at $\boldsymbol{t}$ and $\hat{t}_S$.

Proposition 18 *Let $\boldsymbol{t} \in [0,1]^n$ and let S be a Schur-convex t-conorm that is strictly cone-increasing at $\hat{t}_S \in [S(\boldsymbol{t}, \boldsymbol{c}), 1]$. Then there is a value $\lambda \in [0,1]$ such that the vector $\boldsymbol{t}^*$ with $i \in D$,*

$$t^*{}_i = \begin{cases} \lambda & \text{if } i = \arg\max_{i \in D} t_i, \\ t_i, & \text{otherwise,} \end{cases} \tag{30}$$

is a minimal refined vector for S and the L_1 norm at $\boldsymbol{t}$ and $\hat{t}_S$.

Proof Assume otherwise. Then, using Theorem 3, there must be a refined vector $\hat{\boldsymbol{t}} \neq \boldsymbol{t}^*$ such that $\|\hat{\boldsymbol{t}} - \boldsymbol{t}\|_1 = \|\boldsymbol{t}^* - \boldsymbol{t}\|_1 = \lambda - t_1^{\downarrow}$ but $S(\hat{\boldsymbol{t}}, \boldsymbol{c}) > S(\boldsymbol{t}^*, \boldsymbol{c})$. Let $\pi(i)$ be the permutation in descending order of $\hat{\boldsymbol{t}}$.

Consider any $k \in \{1, ..., n\}$. Then $\sum_{i=1}^{k} t^*{}_i^{\downarrow} = \sum_{i=1}^{k} t_i^{\downarrow} + (\lambda - t_1^{\downarrow})$, while $\sum_{i=1}^{k} \hat{t}_j^{\downarrow} = \sum_{i=1}^{k} t_{\pi(i)} + \sum_{i=1}^{k} (\hat{t}_i - t_{\pi(i)})$. There is no permutation with higher sum than in descending order, so $\sum_{i=1}^{k} t_{\pi(i)} \leq \sum_{i=1}^{k} t_i^{\downarrow}$. Furthermore, since $\|\hat{\boldsymbol{t}} - \boldsymbol{t}\|_1 = \lambda - t_1^{\downarrow}$, $\sum_{i=1}^{k} (\hat{t}_i - t_{\pi(i)}) \leq \lambda - t_1^{\downarrow}$. Therefore, $\sum_{i=1}^{k} \hat{t}_i^{\downarrow} \leq \sum_{i=1}^{k} t^*{}_i^{\downarrow}$, that is, $\boldsymbol{t}^*$ majorizes $\hat{\boldsymbol{t}}$, and by Schur convexity of S, $S(\boldsymbol{t}^*, \boldsymbol{c}) \geq S(\hat{\boldsymbol{t}}, \boldsymbol{c})$. □

C.5 L_1 minimal refinement function for residuums

Proposition 19 *Let $t_1, t_2 \in [0, 1]$ and let T be a strict Schur-concave t-norm with additive generator g. Consider its residuum $R(t_1, t_2) = \sup\{z | T(t_1, z) \leq t_2\}$ that is strictly cone-increasing at $0 < \hat{t}_R \in [R(t_1, t_2), \max_R]$. Then there is a value $\lambda \in [0, 1]$ such that $\boldsymbol{t}^* = [t_1, \lambda]^\top$ is a minimal refined vector for R and the L_1 norm at $\boldsymbol{t}$ and t.*

Proof We will assume $t_1 > t_2$, as otherwise $R(t_1, t_2) = 1$ for any residuum, which necessarily means $\hat{t}_R = 1$ and so $\boldsymbol{t}^* = \boldsymbol{t}$. Assume $\boldsymbol{t}^*$ is not minimal. Since R is strictly cone increasing at $\hat{t}_R$, by Theorem 3[6] there must be some $\hat{\boldsymbol{t}}$ such that $\|\hat{\boldsymbol{t}} - \boldsymbol{t}\| = \|\boldsymbol{t}^* - \boldsymbol{t}\| = \lambda - t_2$ but $R(\hat{t}_1, \hat{t}_2) > R(t^*{}_1, t^*{}_2)$. Since R is non-decreasing in the first argument and non-increasing in the second, we consider $\hat{\boldsymbol{t}} = [t_1 - \epsilon, \lambda - \epsilon]^\top$ for $\epsilon > 0$.

The residuum constructed from continuous t-norms with an additive generator can be computed as $R(t_1, t_2) = g^{-1}(\max(g(t_2) - g(t_1), 0))$. Since we assumed $R(t^*{}_1, t^*{}_2) < R(\hat{t}_1, \hat{t}_2)$, applying g to both sides,

$$\begin{aligned} \max(g(\lambda) - g(t_1), 0) &> \max(g(\lambda - \epsilon) - g(t_1 - \epsilon), 0) \\ g^{-1}(g(\lambda) + g(t_1 - \epsilon)) &< g^{-1}(g(\lambda - \epsilon) + g(t_1)) \\ T(\lambda, t_1 - \epsilon) &< T(\lambda - \epsilon, t_1) \end{aligned}$$

where in the second step we assume $\lambda \leq t_1$, that is, we are not setting new consequent larger than the antecedent, as otherwise we could find a smaller refined vector by setting it to exactly t_1. In the last step we use that T is strict, as then $T(t_1, t_2) = g^{-1}(g(t_1) + t_2))$. We now use the majorization as $\lambda + t_1 - \epsilon = \lambda - \epsilon + \boldsymbol{t}$.

Since $\lambda \leq t_1$, surely $t_1 > \lambda - \epsilon$. Then there are two cases:

1. $\lambda \geq t_1 - \epsilon$. Then $t_1 \geq \lambda$ as assumed.
2. $t_1 - \epsilon \geq \lambda$. Then clearly $\boldsymbol{t} \geq t_1 - \epsilon$ as $\epsilon > 0$.

[6] This theorem has to be adjusted for the fact that fuzzy implications are non-increasing in the first argument. It can be applied by considering $1 - t_1$.

Therefore $[\lambda-\epsilon, t_1]^\top$ majorizes $[\lambda, \boldsymbol{t}-\epsilon]^\top$, and by Schur concavity $T(\lambda, \boldsymbol{t}-\epsilon) \geq T(\lambda-\epsilon, t_1)$ which is a contradiction. □

D Product t-norm with L_2 norm

In this appendix, we consider the refinement functions for the product t-norm under the L_2-norm. We find that there is no simple closed-form parameterization in terms of $\hat{t}_\varphi$, but we can find approximations in linear time. These are satisfactory to reliably find the minimal refinement function.

In the following, we wil ignore constants and consider formulas $\bigwedge_{i=1}^n P_i$, and consider the problem in Eq. 7. We consider the logarithm of the product as its optimum coincides.

$$\begin{aligned} \text{For all} \quad & \boldsymbol{t} \in [0,1]^n, \hat{t}_{T_p} \in [T_P(\boldsymbol{t}), 1] \\ \min_{\hat{\boldsymbol{t}}} \quad & \sum_{i=1}^n (\hat{t}_i - t_i)^2 \\ \text{such that} \quad & \sum_{i=1}^n \log \hat{t}_i = \log \hat{t}_{T_p} \\ & \hat{t}_i - 1 \leq 0 \end{aligned} \tag{31}$$

With Lagrangian $L = \sum_i (\hat{t}_i - t_i)^2 + \lambda(\sum_i \log \hat{t}_i - \log \hat{t}_{T_p}) - \gamma_i(\hat{t}_i - 1)$, and so

$$\begin{aligned} \frac{\partial L}{\partial \hat{t}_i} &= 2(\hat{t}_i - t_i) - \gamma_i + \frac{\lambda}{\hat{t}_i} = 0 \\ \lambda &= (\gamma + 2t_i - 2\hat{t}_i)\hat{t}_i \end{aligned}$$

Since this holds for all i, we find that for all i, j, $(\gamma + 2t_i - 2\hat{t}_i)\hat{t}_i = (\gamma + 2t_j - 2\hat{t}_j)\hat{t}_j = \lambda$. We partition $\{1, ..., n\}$ into sets I and M, where I contains all i such that $\hat{t}_i < 1$, and M those where $\hat{t}_i = 1$. For $i \in I$, by noting that using the complementary slackness condition $\gamma_i = 0$, this induces a quadratic equation in $\hat{t}_i$ with solutions

$$\hat{t}_i = \frac{1}{2}(t_i \pm \sqrt{t_i^2 - 2\lambda}). \tag{32}$$

Since we assume $\hat{t}_i \geq t_i$, we have to take the solution that adds the root of the determinant, that is, $\hat{t}_i = \frac{1}{2}(\sqrt{t_i^2 - 2\lambda} + t_i)$. Furthermore, since we constrain for $i \in I$ that $\hat{t}_i < 1$, we find that

$$\begin{aligned} 1 &> \frac{1}{2}(t_i + \sqrt{t_i^2 - 2\lambda}) \\ 2 - t_i &> \sqrt{t_i^2 - 2\lambda} \\ \lambda &> 2t_i - 2. \end{aligned}$$

Therefore, given some chosen value of c, we require for all $i \in I$ that $\lambda > 2t_i - 2$, and so,

$$\min_{i \in I} 2t_i - 2 > \lambda$$

Unfortunately, finding the exact value of λ such that $T_P(\hat{\boldsymbol{t}}) = \hat{t}_{T_P}$ is a challenge. Filling in $\hat{t}_i$, we find

$$T_P(\hat{\boldsymbol{t}}) = \prod_{i=1}^{n}(\hat{t}_i) = \prod_{i \in I}\frac{1}{2}(t_i + \sqrt{t_i^2 - 2\lambda}) = \hat{t}_{T_P}. \tag{33}$$

This is a $2n$-th degree polynomial in λ, and we were not able to find an obvious, general closed form solution to it. Mathematica (Inc, 2019) finds a complicated closed form formula for $n = 2$, but cannot find closed form formulas for $n > 2$.

We also still need to figure out how to partition $i = \{1, ..., n\}$ into I and M. Since $\hat{t}_i$ as computed by Eq. 32 is a strictly decreasing function in λ for all $i \in I$, we have the following unproven proposition. It supports the result given in Theorem 4.

Proposition 20 *For all* $\lambda \in [\min_{i=1}^{n} 2t_i - 2, 0]$, *the function*

$$\rho^*_{T_P}(\boldsymbol{t}, \lambda)_i = \begin{cases} \frac{1}{2}(t_i + \sqrt{t_i^2 - 2\lambda}) & \text{if } 2t_i - 2 > \lambda, \\ 1 - t_i & \text{otherwise.} \end{cases} \tag{34}$$

has the following properties:

1. $\rho^*_{T_P}(\boldsymbol{t}, \lambda)$ *is a minimal refinement vector for the product t-norm, the* L_2 *norm and* $\hat{t}_{T_P} = T_P(\rho^*_{T_P}(\boldsymbol{t}, \lambda))$;
2. $\hat{t}_{T_P} = T_P(\rho^*_{T_P}(\boldsymbol{t}, \lambda))$ *is a strictly decreasing function in c on* $(\min_{i=1}^{n} 2t_i - 2, 0]$, *and so there is a bijection between* λ *and* $t \in [T_P(\boldsymbol{t}), 1]$ *on this interval.*

The second property is easy to see by noting the derivative of $\rho^*_{T_P}(\boldsymbol{t}, \lambda)$ is negative on $\lambda \in (\min_{i=1}^{n} 2t_i - 2]$, but for the first we do not have a direct proof as of yet and leave this for future work.

Although $\rho^*_{T_P}(\boldsymbol{t}, \lambda)$ is not parameterized in terms of $\hat{t}_{T_P}$, it can still be used in practical scenarios where λ can be seen as the negative "confidence" in the clause. A practical implementation could learn a weight for the clause between 0 and 1, and then transform it to the domain of λ by dividing by $\min_{i=1}^{n} 2t_i - 2$. Alternatively, $\|T_P(\boldsymbol{t} + \rho^*_{T_P}(\boldsymbol{t}, \lambda)) - \hat{t}_{T_P}\|_2$ can be minimized with respect to λ using mathematical optimization methods like gradient descent or Newton's method to find answers in terms of $\hat{t}_{T_P}$.

E Additional experiments

This Appendix presents additional experiments when $\hat{t}_\varphi$ is not 1.

E.1 Results - Refined value 0.3

Figures 10 and 11 present the results when the refined value $\hat{t}_\varphi = 0.3$.

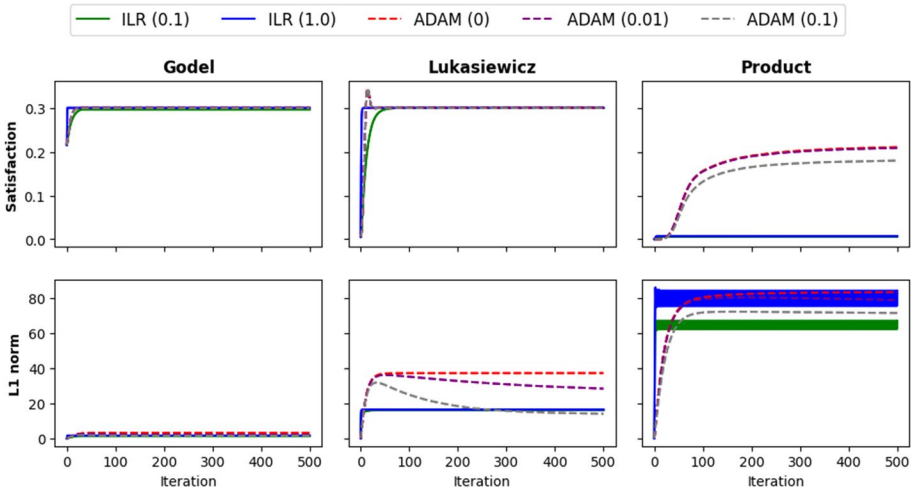

Fig. 10 Comparison of ILR with ADAM on uf20-91 of SATLIB. Refined value 0.3. The x-axis corresponds to the number of iterations, while the y-axis is the value of $\hat{t}_\varphi$ in the first row of the grid and the L_1 norm in the second row

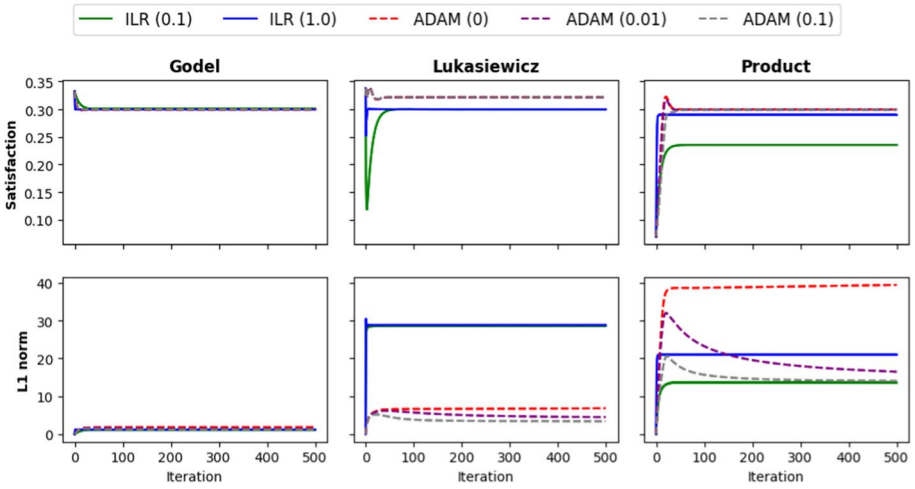

Fig. 11 Comparison of ILR with ADAM on the uf20-91 with 20 clauses. Refined value 0.3

E.2 Results - Refined value 0.5

Figures 12 and 13 present the results when the refined value $\hat{t}_\varphi = 0.5$. We note that the satisfaction for ADAM in Łukasiewicz converges above 0.5 in Fig. 12. This means the final truth value is too high, and it has not found a proper solution here.

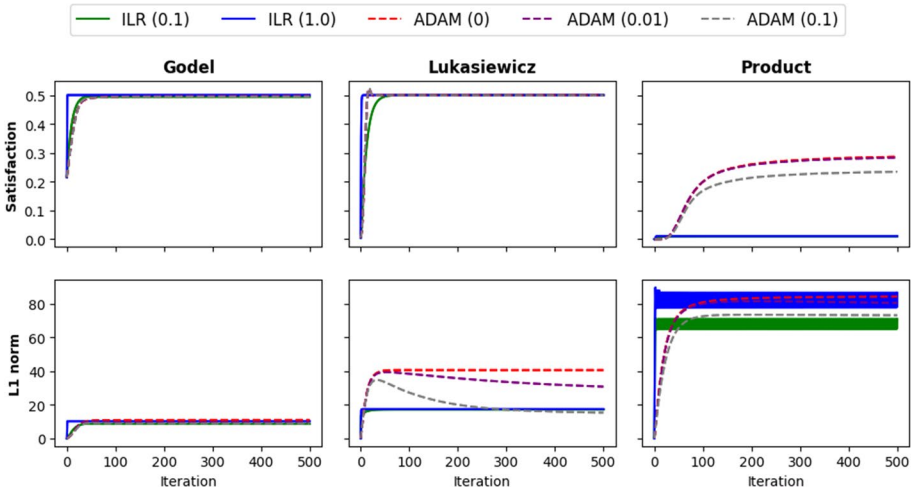

Fig. 12 Comparison of ILR with ADAM on the uf20-91 of SATLIB. Refined value 0.5

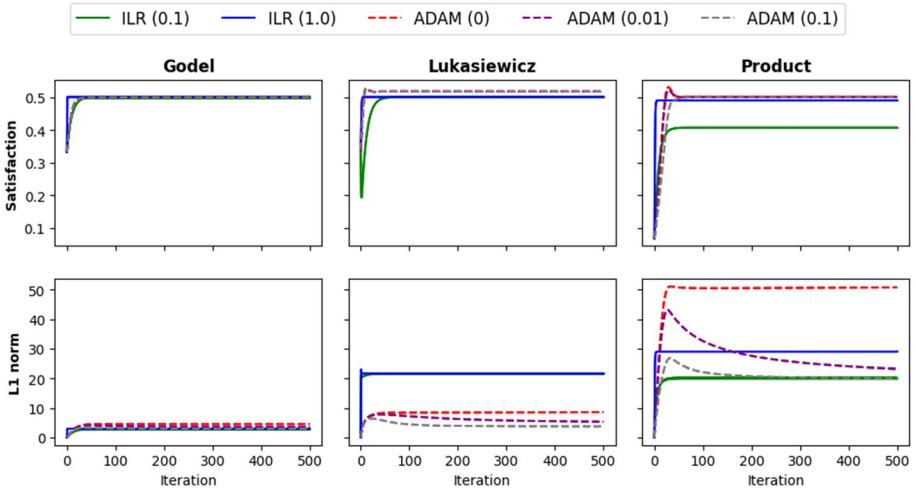

Fig. 13 Comparison of ILR with ADAM on the uf20-91 with 20 clauses. Refined value 0.5

Acknowledgements Alessandro Daniele and Emile van Krieken are involved in a HumaneAI Microproject. HumaneAI received funding from the European Union's Horizon 2020 research and innovation program under Grant Agreement No 761758.

Author Contributions AD and EK: formal proofs, experiments, writing; FH and LS: supervision, writing.

Funding Alessandro Daniele and Emile van Krieken are involved in a HumaneAI Microproject. HumaneAI received funding from the European Union's Horizon 2020 research and innovation program under Grant Agreement No 761758.

Availability of data and material Data used in this work can be downloaded from https://www.cs.ubc.ca/ hoos/SATLIB/benchm.html

Code Availability The code is available as an open-source project on GitHub.https://github.com/DanieleAlessandro/IterativeLocalRefinement

Declarations

Conflict of interest The authors have no conflicts of interest to declare that are relevant to the content of this article

Ethical approval We declare that our manuscript follows the ethics rules provided in https://www.springer.com/gp/editorial-policies/ethical-responsibilities-of-authors.

Consent to participate Not applicable

Consent for publication Not applicable

References

Ahmed, K., Teso, S., Chang, K.-W., den Broeck, G. V., & Vergari, A. (2022) Semantic probabilistic layers for neuro-symbolic learning. CoRR, arXiv:2206.00426.

Alsina, C. (1984) On Schur-Concave t-norms and triangle functions. In: W. Walter, editor, General Inequalities 4: In Memoriam Edwin F. Beckenbach 4th International Conference on General Inequalities, Oberwolfach, May 8–14, 1983, pages 241–248. Birkhäuser, Basel, ISBN 978-3-0348-6259-2. https://doi.org/10.1007/978-3-0348-6259-2_22.

Badreddine, S., d'Avila Garcez, A., Serafini, L., & Spranger, M. (2022). Logic tensor networks. *Artificial Intelligence, 303*, 103649.

Calvo, T., Kolesárová, A., Komorníková, M., & Mesiar, R. (2002) Aggregation operators: Properties, classes and construction methods. In T. Calvo, G. Mayor, and R. Mesiar, editors, Aggregation Operators: New Trends and Applications, pages 3–104. Physica-Verlag HD, Heidelberg, ISBN 978-3-7908-1787-4.

Chowdhery, A., Narang, S., Devlin, J., Bosma, M., Mishra, G., Roberts, A., Barham, P., Chung, H. W., Sutton, C., Gehrmann, S., Schuh, P., Shi, K., Tsvyashchenko, S., Maynez, J., Rao, A., Barnes, P., Tay, Y., Shazeer, N., Prabhakaran, V., Reif, E., Du, N., Hutchinson, B., Pope, R., Bradbury, J., Austin, J., Isard, M., Gur-Ari, G., Yin, P., Duke, T., Levskaya, A., Ghemawat, S., Dev, S., Michalewski, H., Garcia, X., Misra, V., Robinson, K., Fedus, L., Zhou, D., Ippolito, D., Luan, D., Lim, H., Zoph, B., Spiridonov, A., Sepassi, R., Dohan, D., Agrawal, S., Omernick, M., Dai, A. M., Pillai, T. S., Pellat, M., Lewkowycz, A., Moreira, E., Child, R., Polozov, O., Lee, K., Zhou, Z., Wang, X., Saeta, B., Diaz, M., Firat, O., Catasta, M., Wei, J., Meier-Hellstern, K., Eck, D., Dean, J., Petrov, S., & Fiedel, N. (2022) PaLM: Scaling Language modeling with pathways. arXiv:2204.02311.

Clarke, F. H., Stern, R. J., & Wolenski, P. R. (1993) Subgradient Criteria for Monotonicity, The Lipschitz Condition, and Convexity. Canadian Journal of Mathematics, 45(6):1167–1183, Dec. 1993. ISSN 0008-414X, 1496-4279. https://doi.org/10.4153/CJM-1993-065-x.

Daniele, A., & Serafini, L. (2019) Knowledge enhanced neural networks. In A. C. Nayak and A. Sharma, editors, PRICAI 2019: Trends in Artificial Intelligence, pages 542–554, Cham. Springer International Publishing. ISBN 978-3-030-29908-8.

Daniele, A., & Serafini, L. (2022) Knowledge enhanced neural networks for relational domains. arXiv preprint arXiv:2205.15762.

Diligenti, M., Gori, M., & Sacca, C. (2017). Semantic-based regularization for learning and inference. *Artificial Intelligence, 244*, 143–165.

Donadello, I., Serafini, L., & d'Avila Garcez, A. (2017) Logic tensor networks for semantic image interpretation. In IJCAI International joint conference on artificial intelligence, pp. 1596—1602.

Dragone, P., Teso, S., & Passerini, A. (2021) Neuro-symbolic constraint programming for structured prediction. In: A. S. d'Avila Garcez and E. Jiménez-Ruiz, editors, Proceedings of the 15th international workshop on neural-symbolic learning and reasoning as part of the 1st international joint conference on learning & reasoning (IJCLR 2021), Virtual conference, October 25-27, 2021, volume 2986 of CEUR workshop proceedings, pages 6–14. CEUR-WS.org.

Fischer, M., Balunovic, M., Drachsler-Cohen, D., Gehr, T., Zhang, C., & Vechev, M. T.(2019) DL2: Training and querying neural networks with logic. In: K. Chaudhuri and R. Salakhutdinov, editors, Proceedings of the 36th international conference on machine learning, ICML 2019, 9-15, Long Beach, California, USA, volume 97 of Proceedings of machine learning research, pp. 1931–1941. PMLR.

Giannini, F., Diligenti, M., Gori, M., & Maggini, M. (2019). On a convex logic fragment for learning and reasoning. *IEEE Transactions on Fuzzy Systems, 27*(7), 1407–1416. https://doi.org/10.1109/TFUZZ.2018.2879627

Giunchiglia, E., & Lukasiewicz, T. (2021). Multi-label classification neural networks with hard logical constraints. *Journal of Artificial Intelligence Research, 72*, 759–818. https://doi.org/10.1613/jair.1.12850

Giunchiglia, E., Stoian, M., Khan, S., Cuzzolin, F., & Lukasiewicz, T. (2022a) ROAD-R: The autonomous driving dataset with logical requirements. June 2022a.

Giunchiglia, E., Stoian, M. C., & Lukasiewicz, T. (2022) Deep learning with logical constraints. In L. D. Raedt, editor, Proceedings of the thirty-first international joint conference on artificial intelligence, IJCAI 2022, Vienna, Austria, 23-29 , pp. 5478–5485. ijcai.org, 2022b. https://doi.org/10.24963/ijcai.2022/767.

Hoernle, N., Karampatsis, R. M., Belle, V., & Gal, K. (2022) MultiplexNet: Towards fully satisfied logical constraints in neural networks. In: Proceedings of the AAAI conference on artificial intelligence, 36(5):5700–5709. ISSN 2374-3468, 2159-5399. https://doi.org/10.1609/aaai.v36i5.20512.

Hoos, H. H.(2000) SATLIB : An online resource for research on SAT. pp. 1–12.

Inc, W. R. (2019) Mathematica, Version 12.0. 2019. Champaign, IL.

Jayaram, B., & Baczynski, M. (2008). *Fuzzy Implications* (Vol. 231). Berlin: Springer.

Kingma, D. P., & Ba, J. (2015) Adam: A method for stochastic optimization. arXiv:1412.6980 [cs], Jan. 2017. Comment: Published as a conference paper at the 3rd International Conference for Learning Representations, San Diego.

Klement, E.-P, Mesiar, R., & Pap, E. (2000) Triangular Norms, volume 8 of Trends in Logic. Springer. ISBN 978-90-481-5507-1. https://doi.org/10.1007/978-94-015-9540-7.

Klement, E. P, Mesiar, R., & Pap, E. (2004) Triangular norms. Position paper II: General constructions and parameterized families. Fuzzy Sets and Systems, 145(3):411–438. ISSN 01650114. https://doi.org/10.1016/S0165-0114(03)00327-0.

LeCun, Y., & Cortes, C. (2010) MNIST handwritten digit database.

Manhaeve, R., Dumančić, S., Kimmig, A., Demeester, T., & De Raedt, L. (2018). DeepProbLog: Neural probabilistic logic programming. In S. Bengio, H. M. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, & R. Garnett (Eds.), *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, 3–8 December 2018.* Canada: Montréal.

Marshall, A. W., Olkin, I., & Arnold, B. C. (2011) Schur-convex functions. In A. W. Marshall, I. Olkin, and B. C. Arnold, editors, Inequalities: Theory of majorization and its applications, pp. 79–154. Springer, New York, NY. ISBN 978-0-387-68276-1. https://doi.org/10.1007/978-0-387-68276-1_3.

Ramesh, A., Dhariwal, P., Nichol, A., Chu, C., & Chen, M. (2022) Hierarchical text-conditional image generation with CLIP Latents. arXiv:2204.06125.

T. J. U. o. N. M. Ross. Fuzzy logic with engineering applications. 2010. ISBN 978-0-470-74376-8. https://doi.org/10.1002/9781119994374.

Roychowdhury, S., Diligenti, M., & Gori, M. (2021) Regularizing deep networks with prior knowledge: A constraint-based approach. Knowledge-Based Systems, 222:106989. ISSN 0950-7051. https://doi.org/10.1016/j.knosys.2021.106989.

Takači, A. (2005). Schur-concave triangular norms: Characterization and application in pFCSP. *Fuzzy Sets and Systems. An International Journal in Information Science and Engineering, 155*(1), 50–64.

Van Dyke, H. A., Vixie, K. R., & Asaki, T. J. (2013) Cone Monotonicity: Structure Theorem, Properties, and Comparisons to Other Notions of Monotonicity. Abstract and Applied Analysis, 2013:1–8, 2013. ISSN 1085-3375, 1687-0409. https://doi.org/10.1155/2013/134751.

van Krieken, E., Acar, E., & van Harmelen, F. (2022) Analyzing differentiable fuzzy logic operators. Artificial Intelligence, 302:103602. ISSN 0004-3702. https://doi.org/10.1016/j.artint.2021.103602.

Wang, P.-W., Donti, P. L., Wilder, B., & Kolter, J. Z. (2019) SATNet: Bridging deep learning and logical reasoning using a differentiable satisfiability solver. In K. Chaudhuri and R. Salakhutdinov, editors, Proceedings of the 36th international conference on machine learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA, volume 97 of Proceedings of Machine Learning Research, pages 6545–6554. PMLR.

Xu, J., Zhang, Z., Friedman, T., Liang, Y., & den Broeck, G. (2018) A semantic loss function for deep learning with symbolic knowledge. In J. Dy and A. Krause, editors, Proceedings of the 35th international conference on machine learning, volume 80, pages 5502–5511, Stockholmsmässan, Stockholm Sweden, PMLR.

Yang, Z., Lee, J., & Park, C. (2022) Injecting Logical Constraints into Neural Networks via Straight-Through Estimators. In K. Chaudhuri, S. Jegelka, L. Song, C. Szepesvári, G. Niu, and S. Sabato, editors, International conference on machine learning, ICML 2022, 17-23 , Baltimore, Maryland, USA, volume 162 of Proceedings of machine learning research, pp. 25096–25122. PMLR, (2022).