Extending Real Logic with Aggregate Functions

S. Badreddine[1], M. Spranger[1,2]

[1] Sony AI, [2]Sony CSL { badreddine.samy@gmail.com, michael.spranger@gmail.com }

Summary

Objective: To study adding Aggregate Functions to Real Logic and Logic Tensor Networks (LTN) [1]

Contributions:
- Formally adding Aggregate Functions to Real Logic
- Showing the importance of such functions by testing the framework on a food chemistry database and querying food and their nutrients

Real Logic and LTN

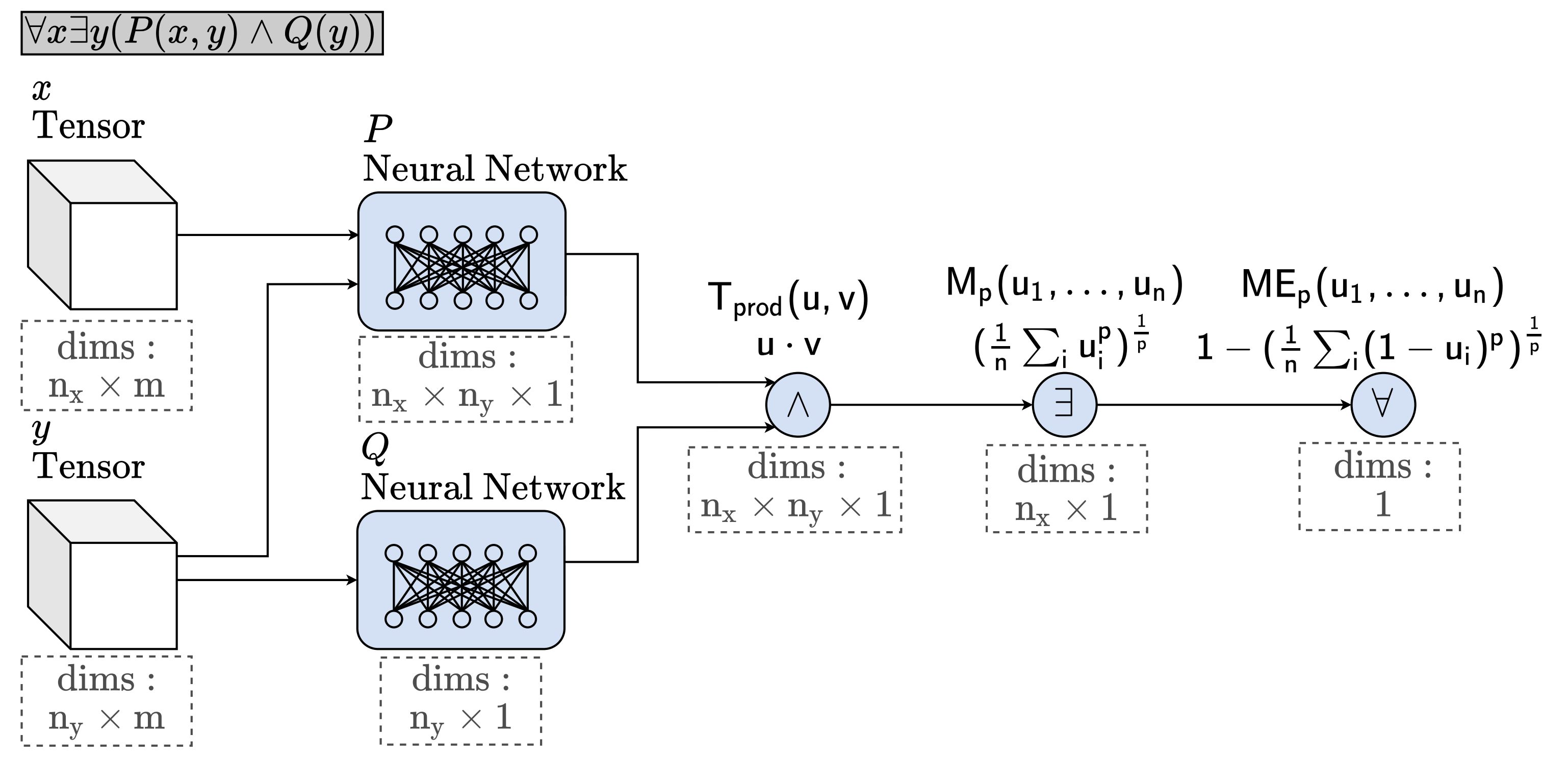
First-Order Language, with fuzzy truth values

Signature:
- Individuals are grounded to real data and features
  - Constants: $alice = [12.6, 5.8, 2.3, \dots]$
  - Variables: $x = \langle alice, bob, charlie \rangle$
- Functions are grounded to real functions. For example, $height(x)$.
- Predicates are grounded as real functions that project in the interval [0,1]. For example, $is\_tall(x)$.

Formulas:
- The connectives $\neg, \wedge, \vee, \rightarrow, \forall, \exists$ are interpreted using fuzzy semantics.
  For example: $p \wedge q = p \cdot q$ $p \vee q = p + q - p \cdot q$
- The evaluation of a formula defines its truth value, its satisfaction score.

Knowledge, Querying and Optimization:
- Knowledge can be represented through
  - Rules: adults are on average taller than kids.
  - Grounding: the height of bob is 170 cm.
- Querying a formula (or term) is simply evaluating the grounding of the formula (or term).
- Groundings can be parametric – e.g., $alice$ is described by a set of features θ to learn. Given a set of rules as a "knowledge-base", one can learn θ such that it maximizes the satisfaction of the rules.

  Because of the fuzzy semantics, Real Logic sentences map to continuous computational graphs that can be derived end-to-end (see below figure). Intuitively, optimizing in Real Logic can be understood as optimizing in a continuous relaxation of traditional First-Order Logic.



A Real Logic sentence grounded to a computational graph

Aggregate Functions

- An aggregate function $A^{(n)} : (D_{in})^n \rightarrow D_{out}$ projects n values from the domain D_{in} to one value in the domain D_{out} . When no confusion can arise, we simply write A instead of $A^{(n)}$.
  For example, with $D_{in} = D_{out} = \mathbb{R}$, **mean**, **std**, or **min**, are common aggregate functions.
- Such aggregators are commonly used in SQL-like queries but are hard to incorporate in traditional logic languages. One approach is to use a 2-sorted logics, where the first sort is interpreted with usual "abstract" semantics, and the second sort denotes real numbers. [2]

In Real Logic

Adding aggregate functions in Real Logic is straightforward:
- Individuals are interpreted with real numbers, on which aggregate functions can immediately operate,
- Variables are grounded as finite sequences of n individuals and constitute the range over which the aggregate function operates.

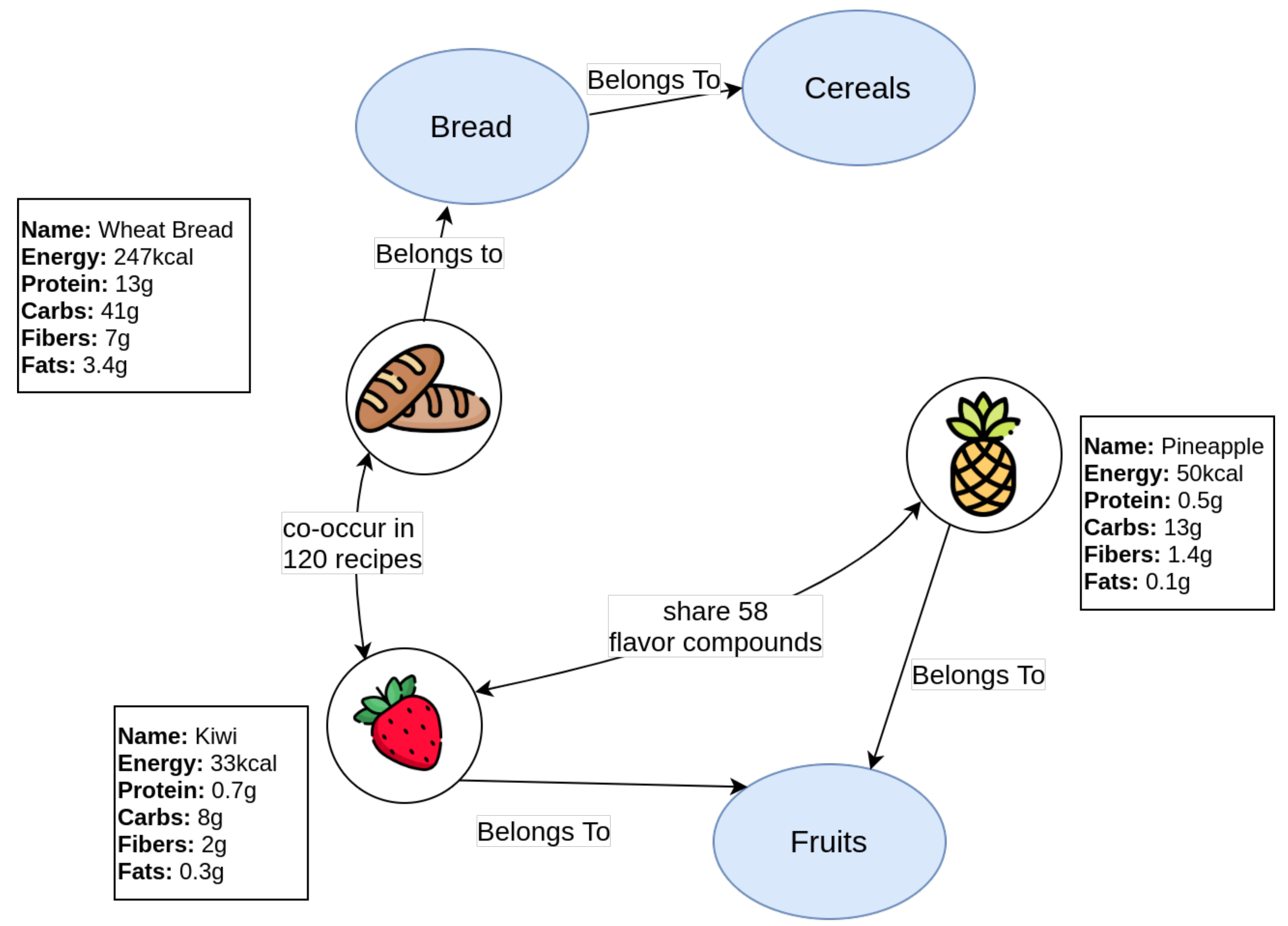
$$x = \langle \text{alice}, \text{bob}, \text{charlie} \rangle$$

$$\begin{aligned} \text{mean}_x \text{height}(x) &= \text{mean}_{i=1}^{3} \text{height}(x)_{(i)} \\ &= \frac{\text{height}(\text{alice}) + \text{height}(\text{bob}) + \text{height}(\text{charlie})}{3} \end{aligned}$$

Experimental Illustration

FooDB [3] Knowledge Graph

- Provides information on food nutrients and food chemistry, including constituents that give food its flavor, color, taste, texture, and aroma.
- We focus on querying macronutrient data from FooDB.



Experimental Illustration (cont.)

Grounding

Symbols	Grounding
Individuals Variable x, Constant $potato$	We ground ingredients using their macronutrient data and their food groups
Functions $kcal(x)$, $prot(x)$	These functional symbols are "getters" of the attributes – e.g., $kcal(x)$ returns the number of calories in x
Aggregate Functions $mean$, std	Typical aggregate functions
Predicates I – Boolean $is\_cereal(x)$	Ontology predicates – e.g., $is\_cereal(x)$ returns 1 if the food group of x is cereal, or 0 else.
Predicates II – Fuzzy $is\_high(v, mean(v), std(v))$	Defines how high a value is using CDFs: that is, the probability of another sample in the variable to be less than or equal to the given value. We approximate the computation using the formula of the CDF for logistic distributions: $sigmoid(\frac{x-\mu}{\sigma})$
Connectives $\neg, \wedge, \vee, \rightarrow, \forall, \exists$	Product configuration presented in [1]

Queries and Results

Querying cereal products and their nutrients in FooDB

id	Query	Top-3 Results	Scores
Q1	$y?$ $\text{is\_cereal}(y) \wedge \neg\text{is\_high}(\text{kcal}(y), \text{mean}_{x:\text{is\_cereal}(x)}\text{kcal}(x), \text{std}_{x:\text{is\_cereal}(x)}\text{kcal}(x))$	Bulgur Wild rice Corn	0.84 0.79 0.79
Q2	$y?$ $\text{is\_cereal}(y) \wedge \text{is\_high}(\text{prot}(y), \text{mean}_{x:\text{is\_cereal}(x)}\text{prot}(x), \text{std}_{x:\text{is\_cereal}(x)}\text{prot}(x))$	Multigrain bread Triticale Potato bread	0.92 0.88 0.84
Q3	$y?$ $\text{Q1}(y) \wedge \text{Q2}(y)$	Multigrain bread Potato bread Oriental wheat	0.56 0.55 0.50
Q4	$\forall y\,(\text{is\_high}(\text{prot}(y), \text{mean}_x\text{prot}(x), \text{std}_x\text{prot}(x)) \rightarrow \text{is\_high}(\text{kcal}(y), \text{mean}_x\text{kcal}(x), \text{std}_x\text{kcal}(x)))$		0.56
Q5	$\forall y\,(\text{is\_high}(\text{fats}(y), \text{mean}_x\text{fats}(x), \text{std}_x\text{fats}(x)) \rightarrow \text{is\_high}(\text{kcal}(y), \text{mean}_x\text{kcal}(x), \text{std}_x\text{kcal}(x)))$		0.74

Conclusions

As a querying system, it combines strength of
- **Descriptive statistics, modeled through fuzzy predicates,**
- **FOL syntax to write complex queries,**
- **SQL-like expressiveness to aggregate and collect insights from data tables.**

Code available on GitHub - https://github.com/logictensornetworks/logictensornetworks

References:

[1] S. Badreddine, A. d. Garcez, L. Serafini, M. Spranger, Logic Tensor Networks, arXiv:2012.13635, 2020.
[2] L. Hella, L. Libkin, J. Nurmonen, L. Wong, Logics with Aggregate Operators, Journal of the ACM 48, pp 880-907, 2001.
[3] www.foodb.ca