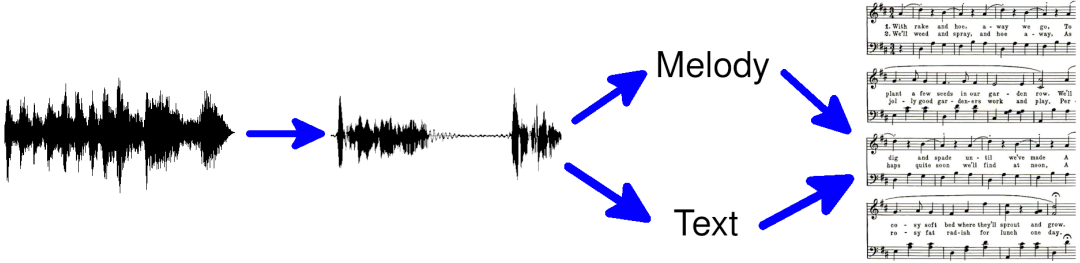
AI-DSL for Autonomous Interoperability

Nil Geisweiller

SingularityNET & OpenCog Foundations

Example: Audio track → Music sheet



Formal Specification of AIs

- Solution:
  - Formal description of AIs
- Challenges:
  - Writing formal description is tedious
  - Requires concepts from real world
  - Mismatches, sophisticated casting
  - Performance evaluation, resource management

Writing Formal Specifications

- Dependent Types: Idris, Agda, Liquid Haskell, etc.
- AIs can help too
  - Code analysis
  - Natural Language Processing of comments

Concepts from the real world

- Ontologies
- Concept creation (AIs)

Mismatches, sophisticated casting

- $h : A \to C$

Mismatches, sophisticated casting

- $h : A \to C$
- We have
  - $f : A \to B$
  - $g : B' \to C$

Mismatches, sophisticated casting

- $h : A \to C$
- We have
  - $f : A \to B$
  - $g : B' \to C$
- $c : B \to B'$

Mismatches, sophisticated casting

- $h : A \to C$
- We have
  - $f : A \to B$
  - $g : B' \to C$
- $c : B \to B'$
- $h = g.c.f$

Performance evaluation, resource management

- Probabilistic Model Checking
- Probabilistic Logic Networks

Evolutionary Programing: Examples of properties

- Deterministic Hillclimbing : (f : Fitness) $\rightarrow$ unimodal(f) $\rightarrow$ converge(f)
- Stochastic Hillclimbing : (f : Fitness) $\rightarrow$ multimodal(f) $\rightarrow$ avg-exp-converge(f)
- BOA : (f : Fitness) $\rightarrow$ decomposable(f) $\rightarrow$ avg-subexp-converge(f)

$\Downarrow$

Probability monads?

Related bibliography:

1. TF-Coder: Program Synthesis for Tensor Manipulations https://arxiv.org/pdf/2003.09040.pdf
2. C2S: Translating Natural Language Comments to FormalProgram Specifications https://www.cs.purdue.edu/homes/lintan/publications/c2s-fse20.pdf
3. Formal Specification for Deep Neural Networks https://www2.eecs.berkeley.edu/Pubs/TechRpts/2018/EECS-2018-25.pdf
4. An epistemic approach to the formal specification of statistical machine learning https://link.springer.com/article/10.1007/s10270-020-00825-2
5. CAMUS: A Framework to Build Formal Specifications for Deep Perception Systems Using Simulators https://arxiv.org/abs/1911.10735
6. VerifAI is a software toolkit for the formal design and analysis of systems that include artificial intelligence (AI) and machine learning (ML) components https://github.com/BerkeleyLearnVerify/VerifAI
7. LTL and Beyond: Formal Languages for Reward Function Specification in Reinforcement Learning https://www.ijcai.org/Proceedings/2019/840
8. Developing Bug-Free Machine Learning Systems With Formal Mathematics https://arxiv.org/abs/1706.08605
9. Verified Stack-Based Genetic Programming via Dependent Types https://cogsys.uni-bamberg.de/events/aaip11/accepted/diehl.pdf

```
Split MOSES into 3 modules:

Vectorize -- Turn syntax tree into vector space (program subspace)
Optimize vector space -- Find good vector candidate
Meta-optimize -- Discover regularities in the vector space

Example: MOSES, evolve syntax trees, call external AI for the
optimization step in vector space.

* MOSES (very abstract) type signature:

moses : Fitness -> Population -> Population

data Candidate = ... -- Syntax tree
type Fitness = Candidate -> Float
type Population = Map Candidate Float

* Optimize Vector Space (very abstract) type signature:

VecFitness -> (Vector Float) -> VecPopulation
type VecFitness = Vector Float -> Float
type VecPopulation = Map (Vector Float) Float

* Meta-optimize (very abstract) type signature:

type OptimizationRecord = Map (Vector Float) Float = VecPopulation
data FitnessEstimate = ... -- Probabilistic model
moptimize : OptimizationRecord -> FitnessEstimate

Run backward from the fitness to the candidates.
```

* Putting it all together:

```
f : Fitness
v : Vectorize
s : Symbolize

vecopt : VecFitness -> (Vector Float) -> VecPopulation
vecopt_rec : VecFitness -> (Vector Float) -> Termination -> VecPopulation

i : Candidate -- Initial candidate
moses f i = vecopt_rec (f . s) (v i) t
  where s, v, t ...

vf : VecFitness
fe : FitnessEstimate
vi : Vector Float
t : Termination
vecopt_rec vf fe vi t = (union sols (vecopt_rec vf nfe (decrease t)))
  where sols = (vecopt vf vi),
        new_fe = (join (moptimize sols) fe (select sols)),
        join = ... -- merge 2 fitness estimates
```