Prometheus AI
Sean Stappas — Supervised by Prof. Vybihal

McGill

Introduction

▶ Prometheus
  ▶ Model of the human brain.
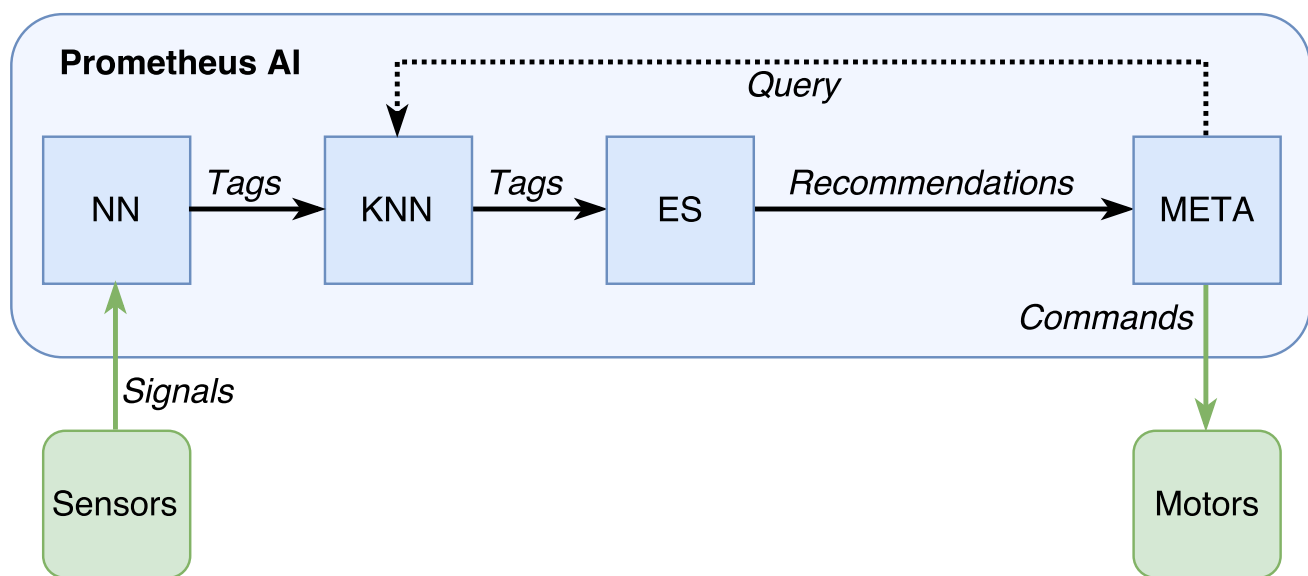 ▶ Controls multiple robots in a swarm.

Figure 1: Prometheus AI model with labeled input and output.

▶ Neural Netowrk (NN)
  ▶ Low-level signal processor.
▶ Knowledge Node Network (KNN)
  ▶ Represents memory.

(a) Direct searching.

(b) Forward searching/thinking.

(c) Backward searching/thinking.
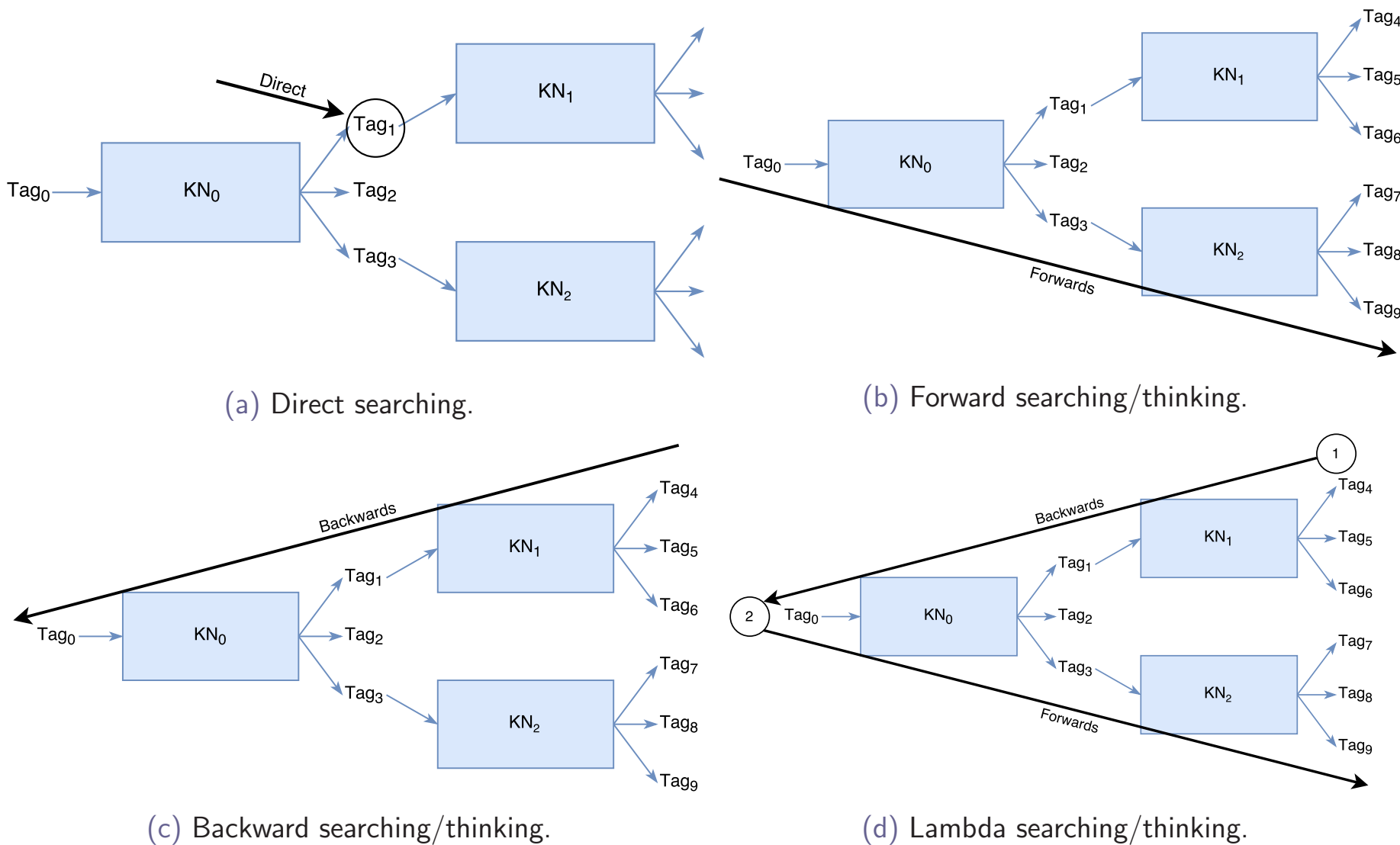
(d) Lambda searching/thinking.

Figure 2: Methods of searching and thinking in the KNN.

▶ Expert System (ES)
  ▶ Logical reasoner.
  ▶ Unaware of context.

Table 1: Fact predicates in the ES.

Fact	Meaning
(A)	A is true or active.
$(A = 1)$	A is equal to 1.
$(A > 1)$	A is greater than 1.
$(A\ ?)$	A can take any value.

$$Fact_1 \cdots Fact_m \to Tag_1 \cdots Tag_n \qquad (1)$$

▶ Meta Reasoner (META)
  ▶ High-level decision-maker.

Problem

▶ Design and implement the ES and KNN in Java.
  ▶ Create an initial design.
  ▶ Build a code skeleton.
  ▶ Implement integration and unit tests.
▶ Supervise undergraduate students working on Prometheus.
  ▶ Provide resources.
  ▶ Review code.

Design & Implementation

▶ Dependencies modeled using Google Guice.
  ▶ Framework for modular dependency injection.
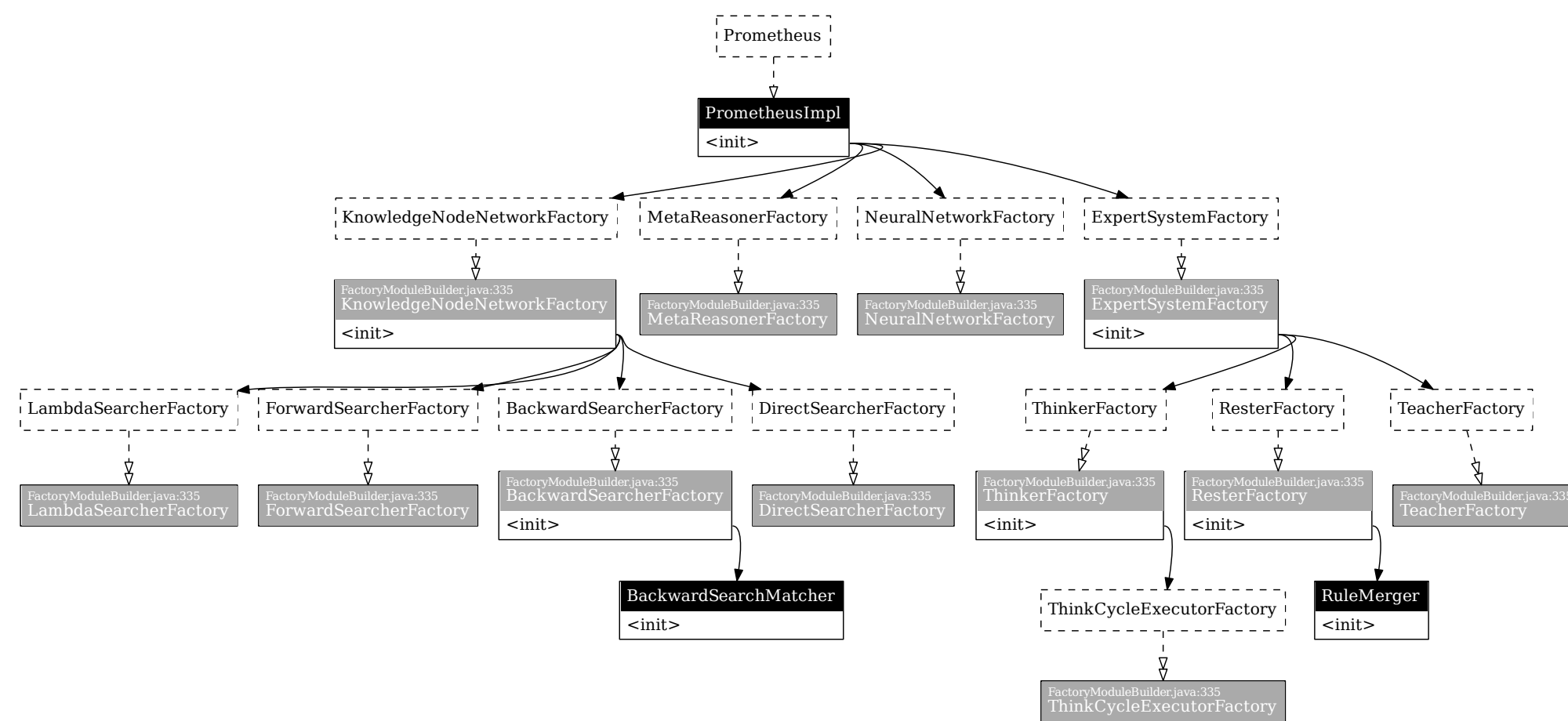 ▶ Allows for easily testable code.

Figure 3: Guice dependency graph.

▶ Tag object is central to the ES and KNN.
  ▶ Represents a unit of information.
  ▶ Can be instantiated as Fact, Rule or Recommendation classes.

(a) Prometheus UML diagram.

(b) ES UML diagram.

(c) Tags UML diagram.
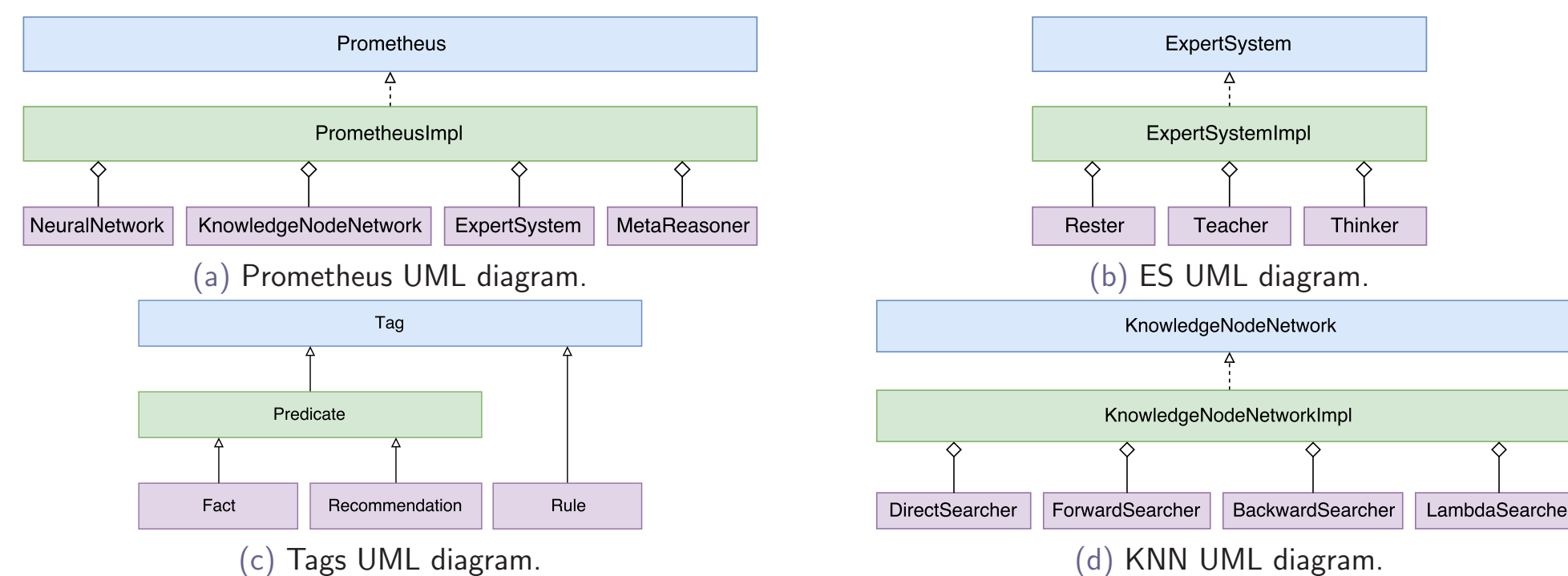
(d) KNN UML diagram.

Figure 4: UML diagrams of the major modules in Prometheus.

Results & Tests

▶ Unit tests.
  ▶ Testing individual methods of every class in the KNN and ES.
  ▶ Dependencies mocked using the Mockito library.
  ▶ 90 % coverage.

▶ Integration tests.
  ▶ Testing end-to-end behavior of the ES and KNN modules.
  ▶ All unit and integration tests written with TestNG and executed with TravisCI.
  ▶ 95 % coverage.

State	Ready Rules	Active Rules	Active Facts	Active Recommendations
Initial	$(A)(B) \to (D)$ $(D)(B) \to (E)$ $(D)(E) \to (F)$ $(G)(A) \to (H)$ $(E)(F) \to (\#Z)$		$(A), (B)$	$(\#X), (\#Y)$
⋮	⋮	⋮	⋮	⋮
Final	$(G)(A) \to (H)$	$(A)(B) \to (D)$ $(D)(B) \to (E)$ $(D)(E) \to (F)$ $(E)(F) \to (\#Z)$	$(A), (B),$ $(D), (E)$ (F)	$(\#X), (\#Y), (\#Z)$

Figure 5: ES test setup representing simple rules and facts that must be brought to quiescence.
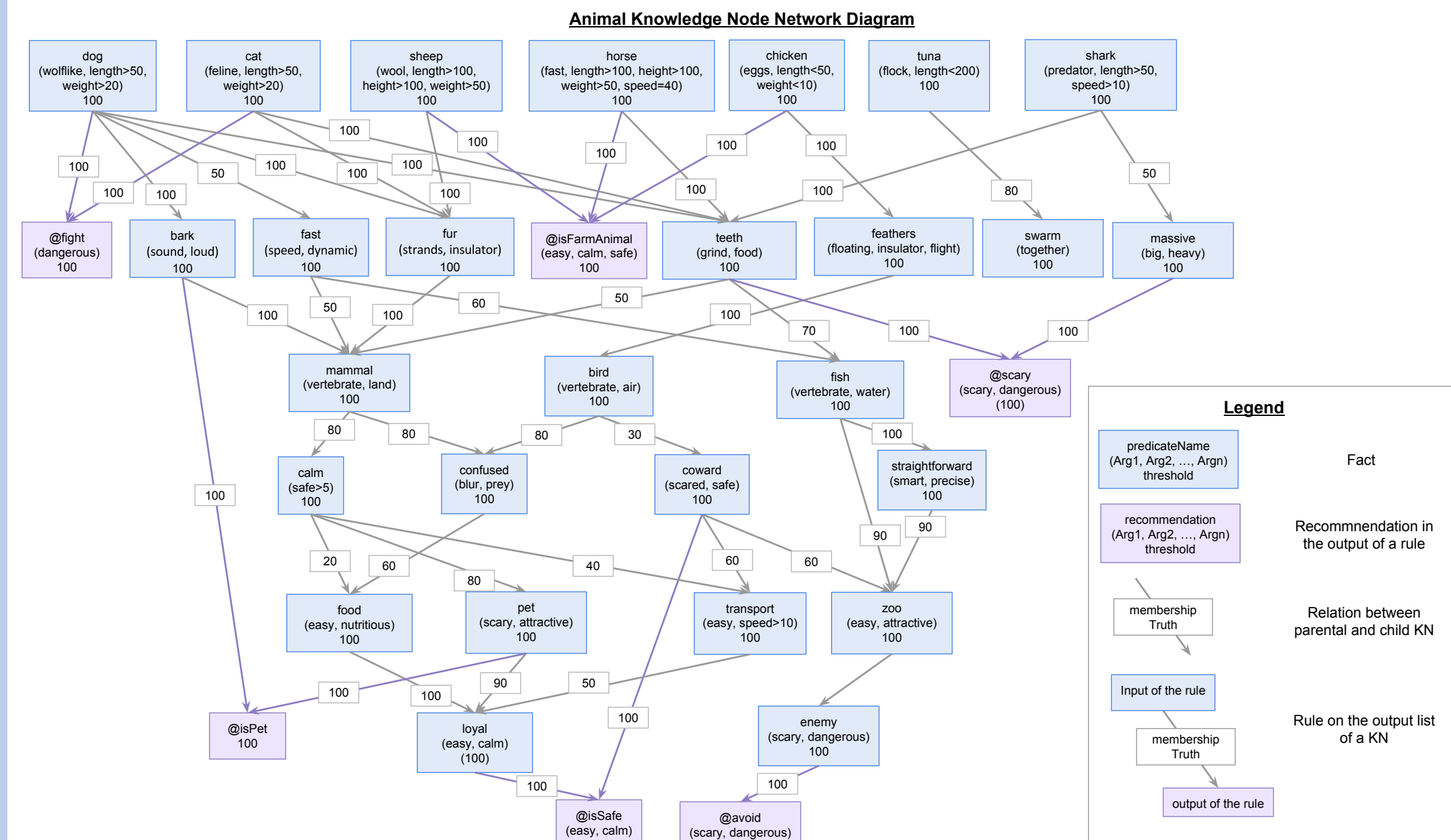
Results & Tests (continued)

Figure 6: Elaborate test KNN network representing connections between memories of animals and their characteristics. Credit for creating the graph goes to volunteer Laurence Liang.

▶ Graph visualization tests.
  ▶ Iterations of the KNN searching algorithms are presented visually.
  ▶ Small nodes are Tags, big nodes are Knowledge Nodes (KNs).
  ▶ Red nodes represent active Tags or fired KNs.
  ▶ For non-active Tags, Facts are purple, Rules are blue and Recommendations are orange.
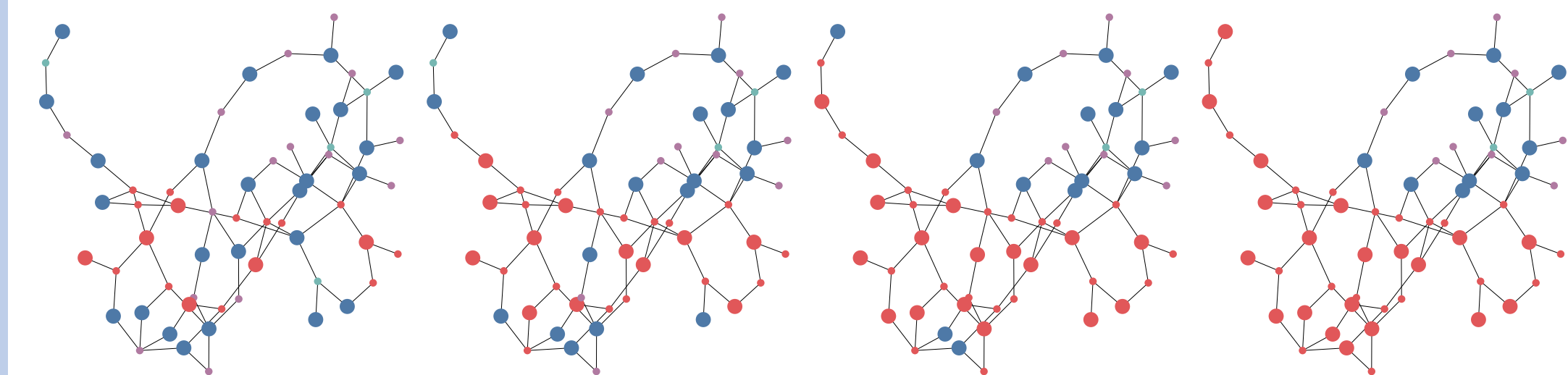

(a) Ply 1.

(b) Ply 2.

(c) Ply 3.

(d) Ply 4.

Figure 7: Forward thinking visualization in the KNN.

(a) Ply 1.

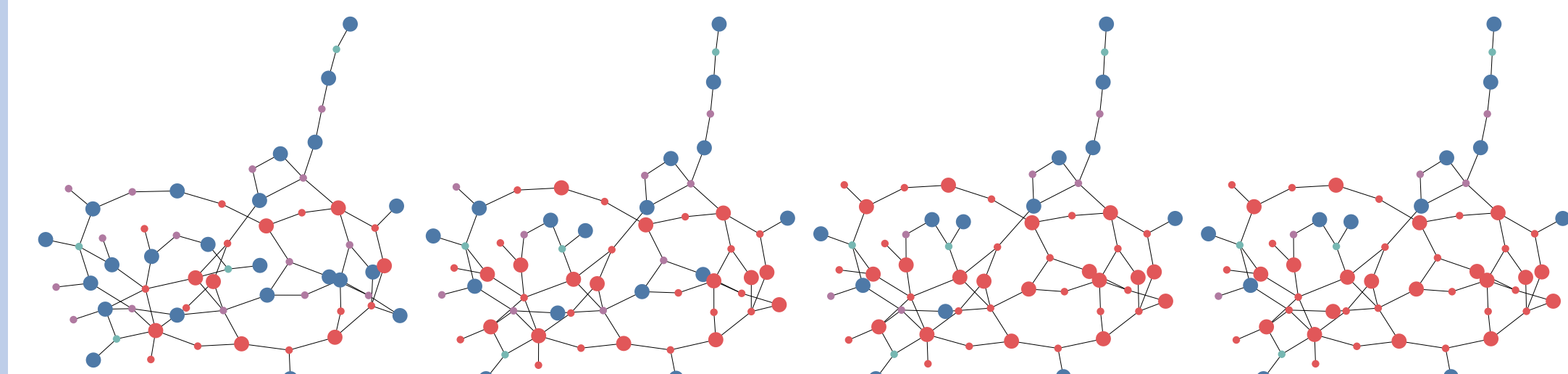(b) Ply 2.

(c) Ply 3.

(d) Ply 4.

Figure 8: Backward thinking visualization in the KNN.

Conclusion

▶ Skills learned:
  ▶ Planning and implementing large software project.
  ▶ Time management.
  ▶ People management.
▶ Possible future work:
  ▶ Implement the missing NN and META layers.
  ▶ Explore further features in the KNN, such as learning and attention.