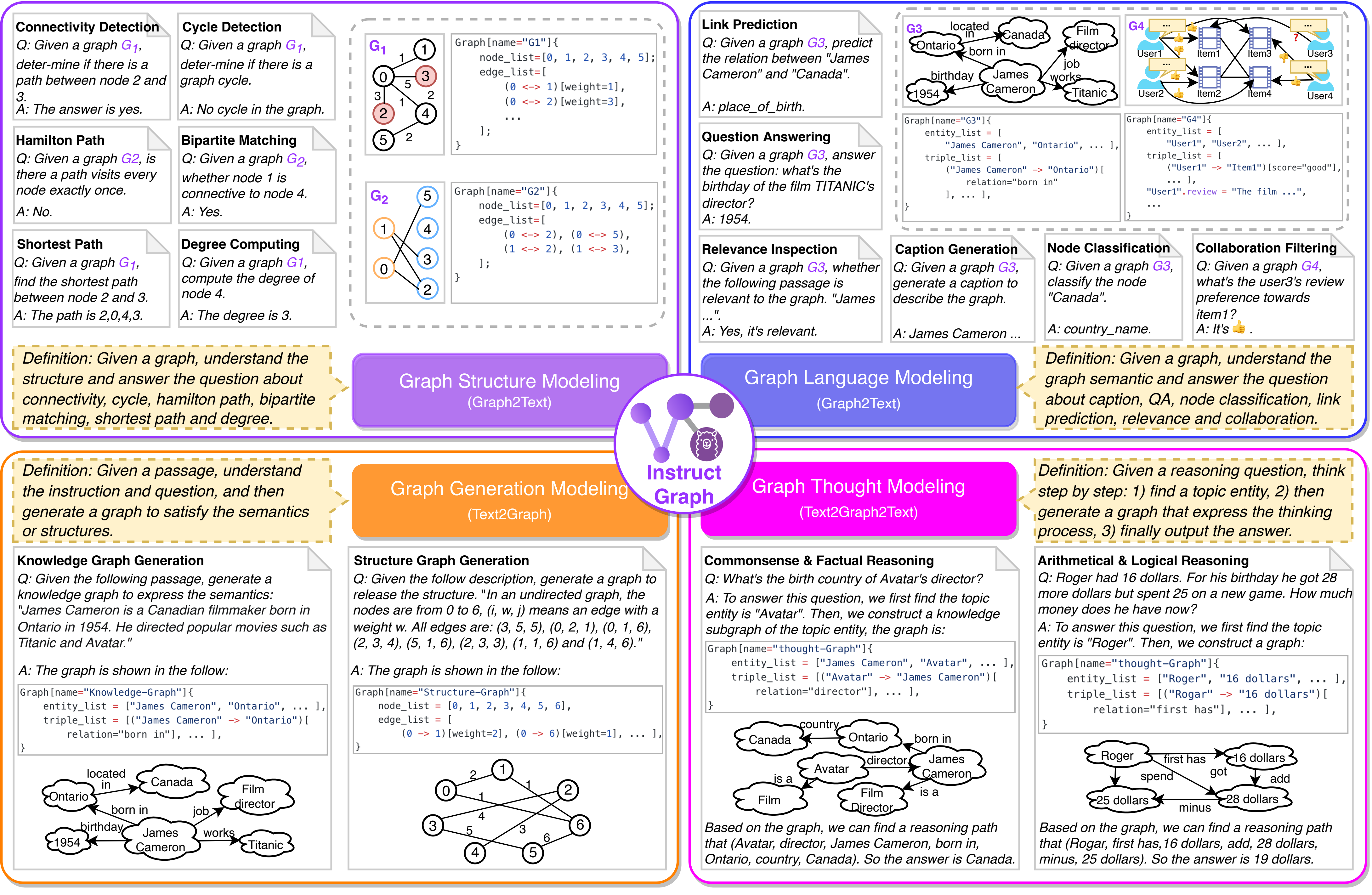

Connectivity Detection

Q: Given a graph G1, deter-mine if there is a path between node 2 and 3.

A: The answer is yes.

Cycle Detection

Q: Given a graph G1, deter-mine if there is a graph cycle.

A: No cycle in the graph.

Hamilton Path

Q: Given a graph G2, is there a path visits every node exactly once.

A: No.

Bipartite Matching

Q: Given a graph G2, whether node 1 is connective to node 4.

A: Yes.

Shortest Path

Q: Given a graph G1, find the shortest path between node 2 and 3.

A: The path is 2,0,4,3.

Degree Computing

Q: Given a graph G1, compute the degree of node 4.

A: The degree is 3.

```
Graph[name="G1"]{
    node_list=[0, 1, 2, 3, 4, 5];
    edge_list=[
        (0 <-> 1)[weight=1],
        (0 <-> 2)[weight=3],
        ...
    ];
}
```

```
Graph[name="G2"]{
    node_list=[0, 1, 2, 3, 4, 5];
    edge_list=[
        (0 <-> 2), (0 <-> 5),
        (1 <-> 2), (1 <-> 3),
    ];
}
```

Definition: Given a graph, understand the structure and answer the question about connectivity, cycle, hamilton path, bipartite matching, shortest path and degree.

Graph Structure Modeling
(Graph2Text)

Link Prediction

Q: Given a graph G3, predict the relation between "James Cameron" and "Canada".

A: place_of_birth.

Question Answering

Q: Given a graph G3, answer the question: what's the birthday of the film TITANIC's director?

A: 1954.

```
Graph[name="G3"]{
    entity_list = [
        "James Cameron", "Ontario", ... ],
    triple_list = [
        ("James Cameron" -> "Ontario")[
            relation="born in"
        ], ... ],
}
```

```
Graph[name="G4"]{
    entity_list = [
        "User1", "User2", ... ],
    triple_list = [
        ("User1" -> "Item1")[score="good"],
        ... ],
    "User1".review = "The film ...",
    ...
}
```

Relevance Inspection

Q: Given a graph G3, whether the following passage is relevant to the graph. "James ...".

A: Yes, it's relevant.

Caption Generation

Q: Given a graph G3, generate a caption to describe the graph.

A: James Cameron ...

Node Classification

Q: Given a graph G3, classify the node "Canada".

A: country_name.

Collaboration Filtering

Q: Given a graph G4, what's the user3's review preference towards item1?

A: It's 👍 .

Graph Language Modeling
(Graph2Text)

Definition: Given a graph, understand the graph semantic and answer the question about caption, QA, node classification, link prediction, relevance and collaboration.

Instruct Graph

Definition: Given a passage, understand the instruction and question, and then generate a graph to satisfy the semantics or structures.

Graph Generation Modeling
(Text2Graph)

Graph Thought Modeling
(Text2Graph2Text)

Definition: Given a reasoning question, think step by step: 1) find a topic entity, 2) then generate a graph that express the thinking process, 3) finally output the answer.

Knowledge Graph Generation

Q: Given the following passage, generate a knowledge graph to express the semantics: "James Cameron is a Canadian filmmaker born in Ontario in 1954. He directed popular movies such as Titanic and Avatar."

A: The graph is shown in the follow:

```
Graph[name="Knowledge-Graph"]{
    entity_list = ["James Cameron", "Ontario", ... ],
    triple_list = [("James Cameron" -> "Ontario")[
        relation="born in"], ... ],
}
```

Structure Graph Generation

Q: Given the follow description, generate a graph to release the structure. "In an undirected graph, the nodes are from 0 to 6, (i, w, j) means an edge with a weight w. All edges are: (3, 5, 5), (0, 2, 1), (0, 1, 6), (2, 3, 4), (5, 1, 6), (2, 3, 3), (1, 1, 6) and (1, 4, 6)."

A: The graph is shown in the follow:

```
Graph[name="Structure-Graph"]{
    node_list = [0, 1, 2, 3, 4, 5, 6],
    edge_list = [
        (0 -> 1)[weight=2], (0 -> 6)[weight=1], ... ],
}
```

Commonsense & Factual Reasoning

Q: What's the birth country of Avatar's director?

A: To answer this question, we first find the topic entity is "Avatar". Then, we construct a knowledge subgraph of the topic entity, the graph is:

```
Graph[name="thought-Graph"]{
    entity_list = ["James Cameron", "Avatar", ... ],
    triple_list = [("Avatar" -> "James Cameron")[
        relation="director"], ... ],
}
```

Based on the graph, we can find a reasoning path that (Avatar, director, James Cameron, born in, Ontario, country, Canada). So the answer is Canada.

Arithmetical & Logical Reasoning

Q: Roger had 16 dollars. For his birthday he got 28 more dollars but spent 25 on a new game. How much money does he have now?

A: To answer this question, we first find the topic entity is "Roger". Then, we construct a graph:

```
Graph[name="thought-Graph"]{
    entity_list = ["Roger", "16 dollars", ... ],
    triple_list = [("Rogar" -> "16 dollars")[
        relation="first has"], ... ],
}
```

Based on the graph, we can find a reasoning path that (Rogar, first has,16 dollars, add, 28 dollars, minus, 25 dollars). So the answer is 19 dollars.