2.3 The Epistemic Challenge of Policy Evaluation

Evaluating policy interventions for AI governance presents unique epistemic challenges that traditional policy analysis methods struggle to address. These challenges arise from the complex causal chains, deep uncertainty, divergent worldviews, and limited empirical grounding that characterize the domain.

Traditional policy analysis relies heavily on historical precedent, empirical data, and established causal models. Cost-benefit analysis quantifies the predicted impacts of interventions based on observed relationships between variables. Scenario planning explores different futures but typically lacks probability estimates. Expert elicitation captures specialist knowledge but often fails to systematically represent interdependencies between factors. None of these approaches fully addresses the specific challenges of AI governance policy evaluation.

Four unique difficulties define the epistemic landscape of AI governance:

First, **complex causal chains with limited empirical grounding** characterize the relationship between governance interventions and risk outcomes. Unlike domains like public health, where interventions have measurable effects on well-defined outcomes, AI governance involves extended causal chains where actions today might influence technological development paths, institutional behaviors, and ultimately risk profiles decades in the future. These chains cannot be empirically tested through traditional methods, yet understanding them is essential for effective governance.

Second, **deep uncertainty about future capability development** creates a challenging environment for prediction. While some aspects of technology evolution follow discernible patterns, transformative capabilities often emerge unexpectedly through conceptual breakthroughs. This uncertainty isn't merely quantitative (what are the error bars on our predictions?) but qualitative (what kinds of capabilities might emerge?), creating fundamental challenges for traditional forecasting methods that rely on extrapolation from past trends.

Third, **divergent worldviews about fundamental risk factors** complicate consensus-building around governance approaches. Experts disagree not just about probability estimates but about which factors matter most and how they relate causally. Some emphasize technical alignment challenges, others focus on competitive dynamics between developers, and still others prioritize institutional oversight mechanisms. Each worldview implies different intervention priorities, yet traditional policy analysis lacks tools for systematically comparing perspectives.

Fourth, **limited opportunities for experimental testing** prevent iterative refinement of governance approaches. Unlike domains where small-scale pilots can test intervention efficacy before wider implementation, many AI governance interventions must be designed without the benefit of experimental evidence. If certain risks materialize only once systems reach advanced capabilities, learning from experience comes too late.

Addressing these challenges requires explicit representation across multiple dimensions:

- **Uncertainty across multiple parameters:** The approach must represent not just uncertainty about outcomes but uncertainty about the relationships between variables and the structure of the causal model itself.
- **Conditional dependencies between variables:** The system needs to capture how different factors influence each other, enabling understanding of complex chains of causation from interventions to outcomes.
- **Comparable representation of different worldviews:** To facilitate productive discourse across perspectives, the approach must represent diverse causal models in a common framework that highlights both agreements and disagreements.
- **Continuous evidence integration mechanisms:** As new information emerges—from theoretical insights, empirical observations, or expert judgments—the system should update its representations to reflect current knowledge.

Historical analogues provide partial insights but no complete template. Nuclear governance established verification protocols and international monitoring, but over a longer timeframe than likely available for AI. Pandemic response developed early warning systems and response protocols, but struggles with similar challenges in predicting novel pathogen emergence. Climate governance demonstrates the difficulty of establishing effective international coordination mechanisms for slow-moving, high-impact risks.

What distinguishes AI governance is the combination of accelerating technological development, distributed creation capability, and potentially irreversible consequences once certain thresholds are crossed. This unique profile necessitates novel approaches to policy evaluation that can handle the epistemic challenges described above while providing actionable insights for governance.

The formal modeling approach developed in this thesis addresses these challenges by making assumptions explicit, facilitating structured comparison of worldviews, and enabling rigorous exploration of intervention impacts across scenarios. By transforming implicit models into explicit representations, it creates a foundation for more productive discourse about governance priorities and approaches, even amid deep uncertainty about future developments.

2.4 Argument Mapping and Formal Representations

Argument mapping provides a bridge between natural language reasoning and formal probabilistic models, enabling the transformation of complex qualitative arguments into structured representations suitable for computational analysis. This section explores two key intermediate representations—ArgDown and BayesDown—that facilitate this transformation process.

Argument maps are structured visualizations that represent the logical relationships between claims, evidence, and objections. Unlike free-form text, they make explicit how different

statements support or challenge one another, forcing clarity about the logical structure of arguments. Traditional argument maps typically include:

- Statements (claims, premises, conclusions) presented as nodes
- Support and attack relationships shown as arrows between nodes
- Hierarchical organization reflecting logical dependencies

These visualizations help identify unstated assumptions, circular reasoning, and gaps in argumentation. However, traditional argument mapping has limited expressivity for representing uncertainty—a crucial element in complex domains like AI risk assessment.

ArgDown extends the concept of argument mapping into a structured text format with a consistent syntax. Developed by Christian Voigt at Karlsruhe Institute of Technology, ArgDown provides a markdown-like notation for representing arguments in a hierarchical structure that can be automatically visualized and analyzed. The basic syntax is:

```
[Statement]: Description of the statement.
 + [Supporting_Statement]: Description of supporting statement.
   + [Further_Support]: Description of additional support.
 - [Opposing_Statement]: Description of opposing statement.
```

For the AMTAIR project, we adapt ArgDown to focus on causal relationships rather than general argumentation, using a modified syntax where the hierarchical structure represents causal influence:

```
[Effect]: Description of effect. {"instantiations": ["effect_TRUE", "effect_FALSE"]}
 + [Cause1]: Description of first cause. {"instantiations": ["cause1_TRUE", "cause1_FALSE"]}
 + [Cause2]: Description of second cause. {"instantiations": ["cause2_TRUE", "cause2_FALSE"]
   + [Root_Cause]: A cause that influences Cause2. {"instantiations": ["root_TRUE", "root_FA
```

This adaptation adds metadata in JSON format to specify possible states (instantiations) of each variable, preparing the structure for probabilistic enhancement. The hierarchical relationships (indented with plus signs) represent causal influence, creating a directed graph structure.

The Rain-Sprinkler-Lawn example in ArgDown format illustrates this structure:

```
[Grass_Wet]: Concentrated moisture on, between and around the blades of grass. {"instantiatio
 + [Rain]: Tears of angles crying high up in the skies hitting the ground. {"instantiations"
 + [Sprinkler]: Activation of a centrifugal force based CO2 droplet distribution system. {"in
   + [Rain]
```

This representation captures the causal structure (both Rain and Sprinkler influence Grass_Wet, and Rain also influences Sprinkler) and specifies the possible states of each variable. However, it lacks probability information, which is where BayesDown extends the representation.

BayesDown builds on ArgDown by adding probability metadata, transforming a purely structural representation into a complete Bayesian network specification. The enhanced format includes:

```
[Node]: Description. {
  "instantiations": ["node_TRUE", "node_FALSE"],
  "priors": {
    "p(node_TRUE)": "0.7",
    "p(node_FALSE)": "0.3"
  },
  "posteriors": {
    "p(node_TRUE|parent_TRUE)": "0.9",
    "p(node_TRUE|parent_FALSE)": "0.4",
    "p(node_FALSE|parent_TRUE)": "0.1",
    "p(node_FALSE|parent_FALSE)": "0.6"
  }
}
```

The Rain-Sprinkler-Lawn example in BayesDown format illustrates this enhancement:

```
[Grass_Wet]: Concentrated moisture on grass. {
  "instantiations": ["grass_wet_TRUE", "grass_wet_FALSE"],
  "priors": {"p(grass_wet_TRUE)": "0.322", "p(grass_wet_FALSE)": "0.678"},
  "posteriors": {
    "p(grass_wet_TRUE|sprinkler_TRUE,rain_TRUE)": "0.99",
    "p(grass_wet_TRUE|sprinkler_TRUE,rain_FALSE)": "0.9",
    "p(grass_wet_TRUE|sprinkler_FALSE,rain_TRUE)": "0.8",
    "p(grass_wet_TRUE|sprinkler_FALSE,rain_FALSE)": "0.0"
  }
}
 + [Rain]: Water falling from the sky. {
   "instantiations": ["rain_TRUE", "rain_FALSE"],
   "priors": {"p(rain_TRUE)": "0.2", "p(rain_FALSE)": "0.8"}
 }
 + [Sprinkler]: Artificial watering system. {
   "instantiations": ["sprinkler_TRUE", "sprinkler_FALSE"],
   "priors": {"p(sprinkler_TRUE)": "0.44838", "p(sprinkler_FALSE)": "0.55162"},
```

```
  "posteriors": {
    "p(sprinkler_TRUE|rain_TRUE)": "0.01",
    "p(sprinkler_TRUE|rain_FALSE)": "0.4"
  }
}
  + [Rain]
```

This representation now contains all the information needed to construct a complete Bayesian network: variables with their possible states, causal relationships between variables, prior probabilities for root nodes, and conditional probability tables for nodes with parents.

The transformation workflow from natural language to BayesDown involves several steps:

1. Identify key variables and their possible states from the text
2. Determine causal relationships between variables
3. Represent the structure in ArgDown format
4. Generate probability questions based on the structure
5. Answer these questions (manually or via LLM)
6. Incorporate probability answers into BayesDown format

This progressive transformation preserves the narrative richness of the original text while adding formal structure. The intermediate representations (ArgDown and BayesDown) remain human-readable, maintaining the connection to the original arguments while enabling computational analysis.

The key innovation in this approach is the separation of structure extraction from probability quantification, which aligns with how experts typically approach complex arguments. First, they identify what factors matter and how they relate causally, then they consider how probable different scenarios are based on those relationships. This two-stage process makes the extraction more robust and the resulting representations more interpretable.

2.5 The MTAIR Framework: Achievements and Limitations

The Modeling Transformative AI Risks (MTAIR) project, led by David Manheim and colleagues, represents a significant precursor to the current research. Launched in 2021, MTAIR aimed to create structured representations of existential risks from advanced AI using Bayesian networks, directed acyclic graphs, and probabilistic modeling. Understanding its achievements and limitations provides important context for the current AMTAIR approach.

MTAIR emerged from the recognition that AI risk discussions often involved complex causal arguments with implicit probability judgments that were difficult to compare or integrate. By formalizing these arguments in structured models, the project sought to make assumptions explicit, enable quantitative analysis, and facilitate more productive discourse across different perspectives on AI risk.

The framework's key innovations included:

1. **Explicit representation of uncertainty through probability distributions:** Rather than presenting point estimates, MTAIR captured uncertainty about parameters using distributions, acknowledging the significant uncertainty in AI risk assessment.
2. **Hierarchical structure for complex scenarios:** The approach used nested models that allowed exploration of different levels of detail, from high-level risk factors to specific technical mechanisms.
3. **Integration of diverse expert judgments:** The framework incorporated perspectives from various specialists, creating a more comprehensive view than any single expert could provide.

The project's practical impact extended beyond its technical achievements. It influenced research prioritization by identifying critical uncertainties that warranted further investigation. It enhanced discourse quality by providing a shared vocabulary and structure for discussing causal pathways to risk. It also created visual representations that made complex arguments more accessible to stakeholders without technical backgrounds.

Despite these achievements, MTAIR faced several important limitations:

1. **Manual labor intensity limiting scalability:** Creating and updating models required substantial expert time, limiting the number and complexity of models that could be developed and maintained. As one team member noted, "It often took several days of work to formalize even relatively straightforward arguments."
2. **Static nature of models once constructed:** The models were essentially snapshots that did not automatically update as new information emerged, requiring manual revision to remain current.
3. **Limited accessibility for non-technical stakeholders:** While visual representations improved accessibility, understanding and interacting with the models still required specialized knowledge.
4. **Challenges in representing multiple worldviews simultaneously:** Comparing different perspectives required creating separate models, making it difficult to identify specific points of agreement and disagreement.

These limitations motivate the current research in automating the extraction and transformation process. As AI capabilities advance and the volume of relevant research grows, manual approaches cannot keep pace with the need for comprehensive, up-to-date models. Automation addresses the scalability limitation by dramatically reducing the time required to create formal representations of expert arguments.

Moreover, incorporating frontier LLMs into the pipeline enables new capabilities that were not feasible in the original MTAIR framework. These include:

1. Processing larger volumes of literature to capture more diverse perspectives
2. Generating intermediate representations that preserve narrative structure
3. Automating the creation of probability questions based on model structure
4. Facilitating integration with live data sources for continuous updates

By building on MTAIR's foundation while addressing its key limitations, the current research maintains continuity with established approaches to AI risk modeling while pushing the boundaries of what's possible through automation and enhanced representation formats.

The evolution from MTAIR to AMTAIR represents a natural progression: as the field matures and the challenges become more pressing, more sophisticated tools are needed to facilitate coordination and decision-making. Automation doesn't replace expert judgment but amplifies it, allowing insights to be captured, formalized, and shared more efficiently across the AI governance community.

3. Own Position and Argument

3.1 The AMTAIR Solution: Automation and Integration

The coordination crisis in AI governance isn't merely a communication problem—it's a fundamental information processing challenge that scales with the complexity of the domain. As AI capabilities advance and research proliferates, even the most diligent experts cannot manually process, integrate, and analyze the growing volume of specialized knowledge. We need computational tools that augment human capabilities, much as telescopes extend our vision beyond natural limits.

AMTAIR—Automating Transformative AI Risk Modeling—represents such a tool. It builds upon the MTAIR framework's conceptual foundation while addressing its core limitations through automation and integration. The approach doesn't replace human judgment but amplifies it, scaling up our collective ability to make implicit models explicit and enabling more rigorous evaluation of governance options.

The system architecture implements a five-stage pipeline that transforms unstructured text into interactive, analyzable models:

1. **Text ingestion and preprocessing:** Source documents enter the system, undergo normalization to handle diverse formats, and are stored with citation information preserved.
2. **LLM-powered extraction:** Documents are analyzed using a two-stage process that first identifies key variables and relationships (represented in ArgDown), then extracts probability information (represented in BayesDown).
3. **Bayesian network construction:** BayesDown representations are transformed into formal Bayesian networks with nodes, edges, and conditional probability tables.

4. **Interactive visualization:** The networks are rendered as interactive visualizations that encode probability information through color and provide progressive disclosure of details.

5. **Analysis and inference:** The system enables sensitivity analysis, intervention modeling, and comparison across worldviews.

What distinguishes AMTAIR from previous approaches is the central role of frontier language models in automating the extraction and transformation processes. Rather than treating these models as black boxes that generate answers, AMTAIR employs them as cognitive partners in a structured workflow, using carefully designed prompts to extract specific types of information and transform it between representations.

Consider how this approach differs from traditional methods of knowledge integration. Typically, synthesizing expert perspectives involves reading papers, taking notes, and mentally constructing a composite view—a process limited by individual cognitive capacity and vulnerable to various biases. AMTAIR externalizes this process, making each step explicit and reproducible. The LLM doesn't determine what's important; it helps transform expert knowledge into structured formats that humans can more easily analyze and compare.

The system's primary innovations lie in three areas:

First, the **two-stage extraction process** separates structural understanding from probability estimation, mirroring how humans typically approach complex arguments. This separation improves extraction quality by focusing LLMs on distinct cognitive tasks and creates interpretable intermediate representations.

Second, the **BayesDown representation format** bridges qualitative and quantitative aspects of arguments, maintaining narrative context while enabling mathematical precision. This hybrid format preserves the connection to original texts while supporting computational analysis.

Third, the **interactive visualization approach** makes complex probabilistic models accessible to non-technical stakeholders through intuitive visual encoding and progressive disclosure of information. This enhances cross-domain communication by creating shared reference points.

These innovations address specific limitations of the MTAIR framework. Where MTAIR required days of expert time to formalize arguments, AMTAIR can process papers in minutes. Where MTAIR created static snapshots, AMTAIR enables dynamic updating through integration with forecasting platforms. Where MTAIR struggled with accessibility, AMTAIR provides intuitive visualizations with multiple levels of detail.

The potential impact extends beyond technical achievements. By making implicit models explicit, AMTAIR helps identify genuine disagreements versus terminological confusion. By enabling systematic comparison across worldviews, it facilitates more productive discourse about

risk factors and interventions. By supporting counterfactual analysis, it allows policymakers to evaluate governance options across diverse scenarios.

This isn't to suggest that computational tools alone can solve the coordination crisis. Human judgment remains essential for interpreting results, contextualizing insights, and making value-laden decisions. But tools like AMTAIR can dramatically enhance our collective ability to process complex information, identify patterns, and evaluate options—capabilities that become increasingly crucial as AI systems grow more powerful and the stakes of governance decisions rise.

3.2 The Two-Stage Extraction Process

The heart of the AMTAIR approach lies in its two-stage extraction process, which transforms unstructured text into structured probabilistic models through distinct steps that mirror human cognitive processes. This separation—extracting structure before probability—creates important advantages for automation quality, intermediate verification, and interpretability.

When humans analyze complex arguments, they typically first determine what factors matter and how they relate causally, then assess how likely different scenarios are based on those relationships. A climate scientist reading a paper first identifies key variables (emissions, warming, effects) and their causal connections before estimating probabilities of outcomes. This natural cognitive sequence inspired AMTAIR's two-stage approach.

Stage 1: Structure Extraction focuses on identifying key variables and their causal relationships from text, transforming unstructured arguments into ArgDown format. This process involves:

1. **Variable identification:** Determining the key factors discussed in the text, including their possible states (e.g., whether a factor is present/absent or has multiple levels)
2. **Relationship mapping:** Establishing how variables influence each other, creating a directed graph of causal connections
3. **Hierarchical organization:** Arranging variables according to their causal relationships, from root causes to final effects
4. **Metadata attachment:** Annotating each variable with its description and possible states in structured JSON format

The LLM prompt for this stage emphasizes clear identification of causal structure without requiring probability judgments, allowing the model to focus entirely on understanding "what affects what" in the text. This specialized prompt includes detailed instructions about ArgDown syntax, examples of well-formed representations, and guidance for preserving the author's intended meaning.

Figure 4 shows a sample of the ArgDown extraction for Carlsmith's model, illustrating how complex qualitative arguments are transformed into structured representations:

[FIGURE 4: Sample ArgDown extraction from Carlsmith's paper showing hierarchical structure of variables related to existential risk]

```
def parse_markdown_hierarchy_fixed(markdown_text, ArgDown=True):
    """
    Parse ArgDown format into a structured DataFrame with parent-child relationships.

    Args:
        markdown_text (str): Text in ArgDown format
        ArgDown (bool): If True, extracts only structure without probabilities
                        If False, extracts both structure and probability information

    Returns:
        pandas.DataFrame: Structured data with node information, relationships, and attribute
    """
    # Clean and prepare the text
    clean_text = remove_comments(markdown_text)

    # Extract basic information about nodes
    titles_info = extract_titles_info(clean_text)

    # Determine hierarchical relationships
    titles_with_relations = establish_relationships_fixed(titles_info, clean_text)

    # Convert to structured DataFrame format
    df = convert_to_dataframe(titles_with_relations, ArgDown)

    # Add derived columns for analysis
    df = add_no_parent_no_child_columns_to_df(df)
    df = add_parents_instantiation_columns_to_df(df)

    return df
```

This key function transforms the ArgDown text into a structured DataFrame, capturing the hierarchical relationships between variables and preparing them for further processing. The function works by identifying node titles, descriptions, and indentation levels, then establishing parent-child relationships based on the hierarchy indicated by indentation.

Stage 2: Probability Integration enhances the structural representation with probability information, creating a complete BayesDown specification. This stage involves:

1. **Question generation:** Automatically creating appropriate probability questions based on the network structure
2. **Probability extraction:** Obtaining probability estimates for each question, either from the text or through LLM inference
3. **Consistency checking:** Ensuring probability distributions sum to 1 and match structural constraints
4. **BayesDown integration:** Incorporating probability information into the ArgDown structure

The key innovation in this stage is the automated generation of appropriate probability questions based on network structure. For each node, the system generates questions about prior probabilities (how likely is this variable in isolation?) and conditional probabilities (how likely is this variable given different states of its parents?).

Figure 5 illustrates how probability questions are derived for a simple node with one parent:

[FIGURE 5: Diagram showing how probability questions are generated based on network structure]

For the "Sprinkler" node with parent "Rain," the system automatically generates questions like:

- What is the probability for Sprinkler=sprinkler_TRUE?
- What is the probability for Sprinkler=sprinkler_TRUE if Rain=rain_TRUE?
- What is the probability for Sprinkler=sprinkler_TRUE if Rain=rain_FALSE?

These questions are then answered either by extracting explicit probabilities from the text or by having the LLM infer reasonable values based on the author's arguments. The answers are structured into a complete BayesDown representation that includes both the causal structure and all necessary probability information.

The visualization below demonstrates the completed extraction for a portion of Carlsmith's model, showing how variables like "Misaligned Power Seeking" are influenced by multiple factors, each with associated probabilities:

[VISUALIZATION: Extracted causal structure from Carlsmith's model with probability information]

This two-stage approach offers several important advantages:

1. **Improved extraction quality:** By focusing on one cognitive task at a time, the LLM performs better at each stage than it would attempting to extract everything simultaneously.
2. **Intermediate verification:** Having ArgDown as an intermediate representation allows human verification before probability extraction, catching structural errors early.

3. **Separation of concerns:** Structure and probability can be updated independently, enabling more flexible maintenance as new information emerges.

4. **Alignment with human cognition:** The process mirrors how experts approach complex arguments, making the system's operation more intuitive and interpretable.

Perhaps most importantly, the intermediate ArgDown representation creates a bridge between qualitative and quantitative aspects of arguments. It preserves the narrative structure and conceptual relationships from the original text while preparing for mathematical precision through probability integration. This hybrid approach maintains the strengths of both worlds: the richness of natural language and the rigor of formal models.

3.3 BayesDown: Bridging Qualitative and Quantitative Representation

If the coordination crisis in AI governance stems partly from incompatible languages across domains—technical researchers speaking in mathematical formalisms, policy specialists in institutional frameworks, and ethicists in normative concepts—then effective coordination requires bridges between these domains. BayesDown serves as such a bridge, combining the narrative richness of qualitative argumentation with the precision of quantitative probability judgments.

Traditional formal representations face a fundamental tradeoff: increase precision and you sacrifice accessibility; enhance accessibility and you lose precision. Mathematical notations offer exactness but exclude many stakeholders. Natural language provides accessibility but permits ambiguity and vagueness. This tradeoff creates communication barriers between technical and policy domains, limiting coordination on complex challenges like AI governance.

BayesDown disrupts this tradeoff by creating a hybrid representation that preserves strengths from both worlds. Its design follows three key principles:

First, **human readability** ensures the representation remains interpretable without specialized training. The syntax builds on familiar conventions from markdown and JSON, maintaining hierarchical relationships through indentation and encapsulating technical details within structured metadata. Unlike purely mathematical notations, the format preserves natural language descriptions alongside formal elements.

Second, **machine processability** enables computational analysis and transformation. The consistent syntax permits automated parsing, formal verification, and conversion to computational models like Bayesian networks. The structured JSON metadata provides clear paths for extracting probability information and mapping it to conditional probability tables.

Third, **contextual preservation** maintains the connection to original arguments. By including descriptive text alongside formal structure, BayesDown retains the narrative context and qualitative considerations that inform probability judgments. This contextual information helps users interpret the model in light of the original arguments.

Consider how these principles manifest in the BayesDown syntax. Each node begins with a bracketed title followed by a natural language description, preserving the core statement being formalized. The JSON metadata contains technical information like instantiations, priors, and posteriors, but keeps this information clearly separated from the narrative content. Hierarchical relationships use indentation and plus symbols, creating a visual structure that mirrors causal influence.

```
[Existential_Catastrophe]: The destruction of humanity's long-term potential due to AI syste
  "instantiations": ["existential_catastrophe_TRUE", "existential_catastrophe_FALSE"],
  "priors": {"p(existential_catastrophe_TRUE)": "0.05", "p(existential_catastrophe_FALSE)":
  "posteriors": {
    "p(existential_catastrophe_TRUE|human_disempowerment_TRUE)": "0.95",
    "p(existential_catastrophe_TRUE|human_disempowerment_FALSE)": "0.0"
  }
}
 + [Human_Disempowerment]: Permanent and collective disempowerment of humanity relative to A
   "instantiations": ["human_disempowerment_TRUE", "human_disempowerment_FALSE"],
   "priors": {"p(human_disempowerment_TRUE)": "0.208", "p(human_disempowerment_FALSE)": "0.79
   "posteriors": {
     "p(human_disempowerment_TRUE|scale_of_power_seeking_TRUE)": "1.0",
     "p(human_disempowerment_TRUE|scale_of_power_seeking_FALSE)": "0.0"
   }
 }
```

This excerpt from the Carlsmith model representation illustrates how BayesDown preserves both the narrative description ("The destruction of humanity's long-term potential…") and the precise probability judgments. Someone without technical background can still understand the core claims and their relationships, while someone seeking quantitative precision can find exact probability values.

The format supports multiple levels of engagement. At the most basic level, readers can follow the hierarchical structure to understand causal relationships between factors. At an intermediate level, they can examine probability judgments to assess the strength of different influences. At the most technical level, they can analyze the complete probabilistic model to perform inference and sensitivity analysis.

This multi-level accessibility creates important advantages for coordination across domains:

1. **Technical-policy translation:** BayesDown provides a common reference point for technical researchers explaining safety concerns and policy specialists evaluating governance options, reducing communication barriers.
2. **Argumentation transparency:** The format makes assumptions explicit, helping identify genuine disagreements versus terminological confusion or unstated premises.

3. **Incremental formalization:** BayesDown supports varying levels of formality, from qualitative structure to complete probability specifications, allowing gradual progression from informal to formal representations.

4. **Verification flexibility:** Human experts can verify extracted representations at different levels—checking structural correctness without assessing probabilities, or focusing on critical probability judgments without reviewing the entire model.

The hybrid nature of BayesDown aligns with how experts typically communicate complex ideas: combining qualitative explanations with quantitative judgments, using natural language to provide context for formal claims, and adjusting precision based on audience needs. By mirroring these natural communication patterns, BayesDown makes formalization more intuitive and accessible.

This bridging function extends beyond representation to influence the entire extraction and analysis workflow. When extracting from text, the two-stage process preserves narrative context alongside formal structure. When visualizing models, interactive interfaces provide both qualitative descriptions and quantitative details. When evaluating policies, counterfactual analysis incorporates both mathematical precision and contextual interpretation.

In the broader context of the coordination crisis, BayesDown demonstrates how thoughtfully designed intermediate representations can overcome communication barriers between domains. Rather than forcing all stakeholders to adopt a single specialized language, it creates a flexible format that accommodates different perspectives while enabling precise analysis—precisely the kind of bridge needed for effective coordination on complex governance challenges.

3.4 Interactive Visualization and Exploration

Complex probabilistic models like Bayesian networks contain rich information, but they often remain inaccessible to many stakeholders. A conditional probability table with dozens of values conveys precise relationships, but few can intuitively grasp its implications. This accessibility gap limits the potential for coordinated action on AI governance challenges—what good is formalization if the resulting models remain opaque to most decision-makers?

AMTAIR addresses this challenge through interactive visualization designed to make complex probabilistic relationships accessible to diverse stakeholders. The approach combines visual encoding of probability information, progressive disclosure of details, and interactive exploration capabilities to create intuitive interfaces for complex models.

The visualization system follows several key design principles:

First, **visual encoding of probability** uses color gradients to represent likelihood values. Nodes are colored on a spectrum from red (low probability) to green (high probability) based on their primary state's probability. This simple visual cue provides immediate insights into which outcomes are more or less likely without requiring numerical interpretation.

Second, **structural classification** uses border colors to indicate node types based on network position. Blue borders designate root causes (nodes without parents), purple borders mark intermediate nodes (with both parents and children), and magenta borders highlight leaf nodes (final effects without children). This classification helps users understand the causal flow through the network.

Third, **progressive disclosure** presents information in layers of increasing detail. Basic node information appears in the visualization itself, additional details emerge in tooltips on hover, and comprehensive probability tables display in modal windows on click. This layered approach prevents information overload while ensuring all details remain accessible.

Fourth, **interactive exploration** allows users to reorganize nodes, zoom in on areas of interest, adjust physics parameters, and investigate probability values. These capabilities transform the visualization from a static image into an explorable knowledge landscape.

Figure 6 shows the interactive visualization of Carlsmith's model, highlighting how color, border styling, and layout work together to represent complex causal relationships:

[FIGURE 6: Interactive visualization of Carlsmith's model showing color-coded nodes and causal relationships]

The visualization system implements these principles through a combination of NetworkX for graph representation and PyVis for interactive display, with custom HTML generation for tooltips and modals:

```
def create_bayesian_network_with_probabilities(df):
    """
    Create an interactive Bayesian network visualization with enhanced probability visualizat
    and node classification based on network structure.
    """
    # Create network structure
    G = nx.DiGraph()

    # Add nodes with attributes
    for idx, row in df.iterrows():
        title = row['Title']
        description = row['Description']
        priors = get_priors(row)
        instantiations = get_instantiations(row)

        G.add_node(title, description=description, priors=priors,
                   instantiations=instantiations, posteriors=get_posteriors(row))

    # Add edges based on parent-child relationships
    for idx, row in df.iterrows():
```

```
        child = row['Title']
        parents = get_parents(row)

        for parent in parents:
            if parent in G.nodes():
                G.add_edge(parent, child)

    # Classify nodes based on network structure
    classify_nodes(G)

    # Create visualization network
    net = Network(notebook=True, directed=True, cdn_resources="in_line",
                  height="600px", width="100%")

    # Configure physics for better layout
    net.force_atlas_2based(gravity=-50, spring_length=100, spring_strength=0.02)
    net.show_buttons(filter_=['physics'])

    # Add graph to network
    net.from_nx(G)

    # Enhance node appearance
    for node in net.nodes:
        node_id = node['id']
        node_data = G.nodes[node_id]

        # Set border color based on node type
        node_type = node_data.get('node_type', 'unknown')
        border_color = get_border_color(node_type)

        # Set background color based on probability
        priors = node_data.get('priors', {})
        background_color = get_probability_color(priors)

        # Create tooltip and expanded content
        tooltip = create_tooltip(node_id, node_data)
        node_data['expanded_content'] = create_expanded_content(node_id, node_data)

        # Set node attributes
        node['title'] = tooltip
        node['label'] = f"{node_id}\np={priors.get('true_prob', 0.5):.2f}"
        node['shape'] = 'box'
```

```
        node['color'] = {
            'background': background_color,
            'border': border_color,
            'highlight': {
                'background': background_color,
                'border': border_color
            }
        }

    # Setup click handling for detailed information
    # [Click handling JavaScript code omitted for brevity]

    return net.show('bayesian_network.html')
```

Beyond the core visualization, the system includes specialized components that enhance understanding of probabilistic relationships:

1. **Probability bars** provide visual representations of probability distributions, showing relative likelihoods of different states using color-coded horizontal bars with numeric labels.
2. **Conditional probability tables** organize complex relationships into structured matrices, displaying how different combinations of parent states influence probability distributions.
3. **Sensitivity indicators** highlight which nodes and relationships most significantly affect outcomes, directing attention to critical factors.

These components work together to create an intuitive interface for complex probabilistic models. A user might start by exploring the overall structure to understand key factors and relationships, hover over nodes of interest to see probability summaries, then click on specific nodes to examine detailed conditional probabilities.

The benefits of this visualization approach extend beyond aesthetic appeal to fundamental improvements in understanding and communication:

First, **intuitive comprehension** of probability relationships becomes possible even for those without formal training in Bayesian statistics. The color coding provides immediate visual cues about which outcomes are more likely, while interactive exploration allows users to develop intuition about how different factors influence results.

Second, **cross-stakeholder communication** improves through shared visual reference points. Technical experts can use the visualizations to explain complex relationships to policy specialists, while governance experts can identify institutional factors that might be incorporated into the models.

Third, **disagreement identification** becomes more precise as stakeholders can point to specific nodes, relationships, or probability values where their views differ, focusing discussion on substantive issues rather than terminological confusion.

Fourth, **intervention assessment** becomes more concrete as users can see how changing specific factors influences downstream effects, providing intuitive understanding of causal pathways and leverage points.

The visualization system demonstrates how thoughtful interface design can overcome barriers to understanding complex formal models. By making probabilistic relationships visually intuitive and progressively disclosing details based on user interest, it creates bridges between mathematical precision and human comprehension—precisely the kind of bridge needed to support coordination across domains in AI governance.

This approach reflects a broader principle: formalization is most valuable when it enhances rather than replaces human understanding. The AMTAIR visualization doesn't simplify complex relationships; it makes them more accessible by leveraging visual cognition, interactive exploration, and progressive disclosure. This human-centered approach to formalization creates tools that augment rather than replace expert judgment, enhancing our collective ability to understand and address complex governance challenges.

3.5 Beyond Extraction: Toward Policy Evaluation

Formalizing expert knowledge through automated extraction creates valuable epistemic infrastructure, but the ultimate goal extends beyond representation to supporting concrete governance decisions. Once implicit models become explicit through the AMTAIR approach, they enable a crucial capability: systematic evaluation of how policy interventions might affect outcomes across different scenarios.

This capability addresses a fundamental challenge in AI governance: making decisions under deep uncertainty about future developments. Traditional approaches often rely on point forecasts or vague qualitative judgments, creating environments where rhetoric outweighs evidence and status determines influence. Formal models enable a more disciplined approach, systematically exploring how different interventions perform across a range of assumptions.

The AMTAIR system supports policy evaluation through three key mechanisms:

First, **counterfactual analysis** implements Pearl's do-calculus to simulate interventions on the causal system. Rather than merely observing correlations, this approach explicitly models what happens when we force a variable to take a specific value, accounting for how this intervention propagates through the causal structure. For example, we can ask how requiring safety demonstrations (setting a variable to a specific value) would affect the likelihood of misaligned systems and ultimately existential risk.

Second, **intervention modeling** provides structured representations of policy options that can be applied to the causal model. Policies are formalized as modifications to specific variables, relationships, or probability distributions, creating concrete representations of how governance actions influence the system. For example, compute governance might be modeled as reducing the probability of rapid capability jumps, while safety standards might increase the likelihood of warning shots.

Third, **cross-worldview comparison** enables evaluation of interventions across different causal models and probability distributions. Rather than assuming a single correct model, this approach acknowledges legitimate uncertainty about causal structure and relationships, testing how interventions perform across different plausible world models. This identifies "robust" policies that work reasonably well regardless of which worldview proves correct—a crucial capability when decisions must be made despite fundamental disagreements.

Consider how these mechanisms apply to Carlsmith's model of existential risk from power-seeking AI. Figure 7 shows the evaluation of a hypothetical governance intervention requiring safety demonstrations before deployment:

[FIGURE 7: Visualization showing policy impact evaluation across Carlsmith model]

The analysis simulates how requiring safety demonstrations affects deployment decisions for potentially misaligned systems, and consequently how this influences the probability of misaligned power-seeking and ultimately existential catastrophe. By comparing the baseline probability (5%) with the intervention probability (3.2% in this example), we can quantify the potential risk reduction from this policy.

The implementation uses counterfactual queries on the Bayesian network:

```
def evaluate_policy_impact(model, intervention_variable, intervention_value, target_variable
    """
    Evaluate the impact of setting a variable to a specific value on a target outcome.

    Args:
        model: Bayesian network model
        intervention_variable: Variable to intervene on
        intervention_value: Value to set for intervention
        target_variable: Outcome variable of interest
        target_value: Outcome value of interest

    Returns:
        dict: Impact analysis including baseline and intervention probabilities
    """
    # Create inference engine
    inference = VariableElimination(model)
```

```
    # Calculate baseline probability
    baseline_query = inference.query(variables=[target_variable])
    baseline_prob = baseline_query.values[baseline_query.state_names[target_variable].index(t

    # Calculate intervention probability using do-calculus
    intervention_query = inference.query(
        variables=[target_variable],
        evidence={intervention_variable: intervention_value},
        do={intervention_variable: intervention_value}  # The do-operation
    )
    intervention_prob = intervention_query.values[intervention_query.state_names[target_varia

    # Calculate impact
    absolute_change = intervention_prob - baseline_prob
    relative_change = absolute_change / baseline_prob * 100 if baseline_prob > 0 else float(

    return {
        'baseline_probability': baseline_prob,
        'intervention_probability': intervention_prob,
        'absolute_change': absolute_change,
        'relative_change': relative_change
    }
```

This function implements the counterfactual analysis, calculating both the baseline probability of the target outcome and the probability after intervention. The `do` operation ensures proper handling of causal effects rather than merely conditioning on observed values.

Beyond analyzing individual interventions, the system can evaluate portfolios of complementary policies, identifying synergies and conflicts between different approaches. For example, it might examine how compute governance, safety standards, and liability rules work together to reduce risk more effectively than any single intervention alone.

The policy evaluation capabilities extend to more sophisticated analyses:

1. **Robustness assessment** examines how sensitive intervention effects are to variations in model parameters, identifying policies that maintain effectiveness despite uncertainty about exact probability values.
2. **Option value analysis** evaluates how different policies affect our ability to gather information and make better decisions in the future, capturing the value of preserving flexibility.
3. **Intervention portfolio construction** identifies sets of complementary policies that address different aspects of risk, creating more robust governance approaches.

4. **Dependency mapping** visualizes prerequisites and enabling conditions between interventions, helping understand sequencing requirements and potential bottlenecks.

These capabilities transform governance discussions from abstract debates about principles to concrete analyses of expected impacts. Rather than merely asserting that a policy would reduce risk, stakeholders can demonstrate specific causal pathways through which the intervention affects outcomes, quantify the magnitude of expected effects, and test robustness across different assumptions.

This approach doesn't eliminate value judgments or normative considerations—those remain essential for determining appropriate governance goals and acceptable tradeoffs. But it adds rigor to instrumental reasoning about how different interventions might achieve those goals, reducing the influence of rhetoric, status, and cognitive biases in policy evaluation.

In the context of the coordination crisis, these policy evaluation capabilities create a shared language for discussing interventions across domains. Technical researchers can express safety concerns in terms of how they affect model variables; policy specialists can formulate governance proposals as interventions on specific factors; ethicists can articulate normative considerations as valued outcomes or constraints on acceptable interventions. This common framework facilitates more productive coordination without requiring all stakeholders to adopt a single specialized vocabulary.

4. Implementation: The AMTAIR Prototype

4.1 System Architecture and Data Flow

The AMTAIR prototype implements the conceptual architecture described earlier through a modular, extensible system designed to transform text into interactive Bayesian networks. This section details the technical realization of this architecture, explaining how different components interact to enable automated extraction and analysis.

At its core, the system consists of five main components connected in a sequential pipeline with feedback loops:

1. **Text ingestion and preprocessing** handles the initial transformation of source documents into a standardized format suitable for extraction. This component supports various input formats (PDF, markdown, plain text) and preserves citation information to maintain provenance.
2. **LLM-powered extraction pipeline** implements the two-stage process for transforming normalized text into structured representations. The first stage extracts structural information (ArgDown), while the second stage enhances it with probability information (BayesDown).

3. **Bayesian network construction** converts BayesDown representations into formal Bayesian networks with nodes, edges, and conditional probability tables. This component includes data transformation, network analysis, and enhancement with derived metrics.

4. **Visualization and interaction interface** creates interactive presentations of the Bayesian networks with probability encoding, progressive disclosure, and exploration capabilities. This component generates HTML with embedded JavaScript for interactivity.

5. **Analysis and inference engine** enables probabilistic reasoning about the networks, including marginal and conditional probability calculations, sensitivity analysis, and counterfactual evaluation for policy assessment.

Figure 8 illustrates the data flow between these components:

[FIGURE 8: Diagram showing data flow between system components]

The implementation uses a combination of Python libraries for different aspects of the pipeline:

- **pandas** for structured data manipulation throughout the pipeline
- **networkx** for graph representation and analysis
- **pgmpy** for Bayesian network construction and inference
- **pyvis** for interactive network visualization
- **requests** for API calls to language models
- **matplotlib** for static visualizations

This architecture balances several design principles:

Modularity ensures that each component can be developed, tested, and improved independently. For example, the extraction pipeline can be enhanced without modifying the visualization system, and different visualization approaches can be implemented without changing the extraction logic.

Explicitness makes the transformation process transparent and inspectable at each stage. Rather than using end-to-end black-box processing, the system creates intermediate representations (ArgDown, BayesDown, DataFrames) that can be examined and verified.

Interactivity prioritizes human engagement with the results, creating rich interfaces that reveal both structural and probabilistic information through visual encoding and progressive disclosure.

Extensibility supports incremental enhancement through well-defined interfaces between components. New capabilities can be added without redesigning the entire system, enabling gradual improvement over time.

The core code organization reflects this architecture:

```
amtair/
    ingestion/             # Text preprocessing and normalization
        pdf_processor.py
        markdown_processor.py
        text_normalizer.py
    extraction/            # LLM-powered extraction pipeline
        argdown_extractor.py
        bayesdown_enhancer.py
        prompt_templates.py
    network/               # Bayesian network construction
        network_builder.py
        data_transformer.py
        metrics_calculator.py
    visualization/         # Interactive visualization
        network_visualizer.py
        html_generator.py
        color_mapper.py
    analysis/              # Analysis and inference
        inference_engine.py
        sensitivity_analyzer.py
        policy_evaluator.py
    utils/                 # Shared utilities
        data_structures.py
        file_operations.py
        logging_config.py
```

This organization makes dependencies explicit while enabling independent development of different components. For example, the extraction team can enhance prompt templates without affecting the network construction code, and the visualization team can improve the user interface without modifying the underlying data structures.

The prototype implementation focused on demonstrating the core pipeline functionality rather than building a complete production system. As a result, the current version has certain limitations:

1. It relies on external API calls to frontier LLMs rather than deploying models locally.
2. It processes documents one at a time rather than ingesting entire literature repositories.
3. It implements basic policy evaluation capabilities without the full range of analysis features.
4. It focuses on BayesDown as the intermediate representation without supporting alternative formats.

Despite these limitations, the prototype successfully demonstrates the feasibility of automating the extraction and transformation process, creating a foundation for more sophisticated implementations in the future.

The architecture's design anticipates future extensions, including integration with prediction markets for dynamic updating, support for cross-worldview comparison, and enhanced policy evaluation capabilities. These extensions would build on the existing foundation rather than requiring architectural redesign, demonstrating the value of the modular approach.

4.2 The Rain-Sprinkler-Lawn Implementation

Before applying the AMTAIR approach to complex real-world risk assessments, I validated the implementation using the canonical rain-sprinkler-lawn example introduced earlier. This simple but complete example allows step-by-step verification of each component in the pipeline, from initial representation to interactive visualization.

The rain-sprinkler-lawn scenario has become something of a "Hello World" for Bayesian networks—simple enough to understand intuitively but complex enough to demonstrate conditional independence and inference. It involves three variables: Rain (whether it's raining), Sprinkler (whether the sprinkler is on), and Grass_Wet (whether the grass is wet). Both rain and the sprinkler can cause the grass to be wet, while rain also influences whether the sprinkler is used (as people typically don't run sprinklers when it's already raining).

Stage 1: ArgDown Representation captures the structural relationships between these variables without probability information. The implementation starts with this representation:

```
[Grass_Wet]: Concentrated moisture on, between and around the blades of grass. {"instantiatio
 + [Rain]: Tears of angles crying high up in the skies hitting the ground. {"instantiations"
 + [Sprinkler]: Activation of a centrifugal force based CO2 droplet distribution system. {"in
   + [Rain]
```

This ArgDown representation captures several key aspects of the scenario:

- The three variables with their natural language descriptions
- Their possible states (TRUE/FALSE for each variable)
- The causal structure (both Rain and Sprinkler influence Grass_Wet, and Rain influences Sprinkler)

The system processes this representation with the parsing function shown in the previous section, transforming it into a structured DataFrame that explicitly represents parent-child relationships:

```
# Process the ArgDown representation
argdown_df = parse_markdown_hierarchy_fixed(argdown_text, ArgDown=True)

# Display the results
print(argdown_df[['Title', 'Description', 'Parents', 'Children', 'instantiations']])
```

This processing correctly extracts the structural information, identifying that:

- Grass_Wet has parents Rain and Sprinkler, but no children
- Rain has no parents, but is a parent to both Grass_Wet and Sprinkler
- Sprinkler has parent Rain and child Grass_Wet

Stage 2: BayesDown Enhancement adds probability information to the structural representation. The implementation first generates appropriate probability questions based on the network structure:

```
# Generate probability questions based on network structure
df_with_questions = generate_argdown_with_questions(argdown_df, "ArgDown_WithQuestions.csv")

# Display sample questions for the Sprinkler node
sprinkler_questions = df_with_questions.loc[df_with_questions['Title'] == 'Sprinkler', 'Gener
print(json.loads(sprinkler_questions))
```

For the Sprinkler node, this generates questions like:

- What is the probability for Sprinkler=sprinkler_TRUE?
- What is the probability for Sprinkler=sprinkler_TRUE if Rain=rain_TRUE?
- What is the probability for Sprinkler=sprinkler_TRUE if Rain=rain_FALSE?

After answering these questions (manually or via LLM), the system incorporates the probability information into a complete BayesDown representation:

```
[Grass_Wet]: Concentrated moisture on, between and around the blades of grass. {
  "instantiations": ["grass_wet_TRUE", "grass_wet_FALSE"],
  "priors": {"p(grass_wet_TRUE)": "0.322", "p(grass_wet_FALSE)": "0.678"},
  "posteriors": {
    "p(grass_wet_TRUE|sprinkler_TRUE,rain_TRUE)": "0.99",
    "p(grass_wet_TRUE|sprinkler_TRUE,rain_FALSE)": "0.9",
    "p(grass_wet_TRUE|sprinkler_FALSE,rain_TRUE)": "0.8",
    "p(grass_wet_TRUE|sprinkler_FALSE,rain_FALSE)": "0.0"
  }
}
```

```
+ [Rain]: Tears of angles crying high up in the skies hitting the ground. {
  "instantiations": ["rain_TRUE", "rain_FALSE"],
  "priors": {"p(rain_TRUE)": "0.2", "p(rain_FALSE)": "0.8"}
}
+ [Sprinkler]: Activation of a centrifugal force based CO2 droplet distribution system. {
  "instantiations": ["sprinkler_TRUE", "sprinkler_FALSE"],
  "priors": {"p(sprinkler_TRUE)": "0.44838", "p(sprinkler_FALSE)": "0.55162"},
  "posteriors": {
    "p(sprinkler_TRUE|rain_TRUE)": "0.01",
    "p(sprinkler_TRUE|rain_FALSE)": "0.4"
  }
}
  + [Rain]
```

This BayesDown representation now contains complete probability information:

- Prior probabilities for each variable (e.g., P(Rain=TRUE) = 0.2)
- Conditional probabilities for variables with parents (e.g., P(Sprinkler=TRUE|Rain=TRUE) = 0.01)

Stage 3: Bayesian Network Construction transforms the BayesDown representation into a formal Bayesian network with nodes, edges, and conditional probability tables. The implementation extracts the information into a structured DataFrame, then converts this into a network representation:

```python
# Extract data from BayesDown representation
extracted_df = parse_markdown_hierarchy_fixed(bayesdown_text, ArgDown=False)

# Enhance the data with calculated metrics
enhanced_df = enhance_extracted_data(extracted_df)

# Create a Bayesian network from the extracted data
def create_bayesian_network(df):
    # Create network structure
    model = BayesianNetwork()

    # Add nodes and edges
    for idx, row in df.iterrows():
        title = row['Title']
        parents = row['Parents'] if isinstance(row['Parents'], list) else []

        # Add node
        model.add_node(title)
```

```
        # Add edges from parents to this node
        for parent in parents:
            model.add_edge(parent, title)

    # Add CPDs for each node
    for idx, row in df.iterrows():
        title = row['Title']
        parents = row['Parents'] if isinstance(row['Parents'], list) else []
        instantiations = row['instantiations'] if isinstance(row['instantiations'], list) els
        priors = row['priors'] if isinstance(row['priors'], dict) else {}
        posteriors = row['posteriors'] if isinstance(row['posteriors'], dict) else {}

        # Create CPD based on whether node has parents
        if not parents:  # No parents - use prior probabilities
            # Implementation details omitted for brevity
        else:  # Has parents - use conditional probabilities
            # Implementation details omitted for brevity

        # Add CPD to model
        model.add_cpds(cpd)

    # Check model validity
    model.check_model()

    return model

# Create the network
bayesian_network = create_bayesian_network(enhanced_df)
```

The resulting Bayesian network correctly represents the causal structure and probability distributions from the BayesDown representation. This network enables various types of probabilistic inference, such as calculating the probability of rain given that the grass is wet:

```
# Create inference engine
inference = VariableElimination(bayesian_network)

# Calculate P(Rain=TRUE | Grass_Wet=TRUE)
result = inference.query(variables=['Rain'], evidence={'Grass_Wet': 'grass_wet_TRUE'})
print(f"P(Rain=TRUE | Grass_Wet=TRUE) = {result.values[0]:.3f}")
```

Visual Result The implementation creates an interactive visualization of the network using the function described in the previous section:

```
# Create interactive visualization
visualization = create_bayesian_network_with_probabilities(enhanced_df)
display(visualization)
```

Figure 9 shows the resulting visualization with color-coded nodes indicating probability values:

[FIGURE 9: Interactive visualization of the rain-sprinkler-lawn Bayesian network]

The visualization correctly encodes the causal structure (arrows from causes to effects) and probability information (node colors indicating likelihood), providing an intuitive representation of the relationships between variables.

Validation To verify the implementation's correctness, I compared computational results from the network with analytical solutions calculated by hand. For example, the probability of wet grass can be calculated analytically:

P(W=TRUE) = , P(W=TRUE|R=r,S=s) × P(R=r) × P(S=s|R=r)

Where the sum is over all possible values of r and s. The computational result from the Bayesian network (0.322) matched the analytical calculation, confirming the implementation's correctness.

Similarly, posterior probabilities like P(R=TRUE|W=TRUE) were verified against analytical calculations using Bayes' rule:

P(R=TRUE|W=TRUE) = P(W=TRUE|R=TRUE) × P(R=TRUE) / P(W=TRUE)

The rain-sprinkler-lawn implementation demonstrates the complete AMTAIR pipeline functioning correctly on a simple but non-trivial example. Each step in the process—from ArgDown representation through BayesDown enhancement to Bayesian network construction and visualization—performs as expected, transforming a structured representation into an interactive, analyzable model.

This validation provides confidence that the approach can be successfully applied to more complex, real-world scenarios like Carlsmith's model of existential risk, which follows the same principles but involves many more variables and relationships.

4.3 Application to Carlsmith's Model

Having validated the implementation on the canonical rain-sprinkler-lawn example, I applied the AMTAIR approach to a substantially more complex real-world case: Joseph Carlsmith's model of existential risk from power-seeking AI. This application demonstrates the system's ability to handle sophisticated multi-level arguments with numerous variables and relationships.

Carlsmith's analysis involves dozens of factors organized in a complex causal structure, from root causes like "Advanced AI Capability" and "Instrumental Convergence" through intermediate factors like "APS Systems" and "Misaligned Power Seeking" to final outcomes like "Existential Catastrophe." The model exhibits several challenging features:

1. **Multi-level structure** with causal chains spanning multiple steps
2. **Divergent pathways** where factors influence outcomes through multiple routes
3. **Complex conditional dependencies** with variables influenced by multiple parents
4. **Variables with three or more possible states** rather than simple binary outcomes
5. **Interconnected clusters** where factors form distinct but related argument groups

The extraction process began with an ArgDown representation capturing the structural relationships between variables:

```
[Existential_Catastrophe]: The destruction of humanity's long-term potential due to AI syste
- [Human_Disempowerment]: Permanent and collective disempowerment of humanity relative to AI
    - [Scale_Of_Power_Seeking]: Power-seeking by AI systems scaling to the point of permanen
        - [Misaligned_Power_Seeking]: Deployed AI systems seeking power in unintended and hig
            - [APS_Systems]: AI systems with advanced capabilities, agentic planning, and sti
                - [Advanced_AI_Capability]: AI systems that outperform humans on tasks that g
                - [Agentic_Planning]: AI systems making and executing plans based on world mo
                - [Strategic_Awareness]: AI systems with models accurately representing power
            - [Difficulty_Of_Alignment]: It is harder to build aligned systems than misaligne
                - [Instrumental_Convergence]: AI systems with misaligned objectives tend to s
                - [Problems_With_Proxies]: Optimizing for proxy objectives breaks correlation
                - [Problems_With_Search]: Search processes can yield systems pursuing differe
            - [Deployment_Decisions]: Decisions to deploy potentially misaligned AI systems.
                - [Incentives_To_Build_APS]: Strong incentives to build and deploy APS syste
                    - [Usefulness_Of_APS]: APS systems are very useful for many valuable tasl
                    - [Competitive_Dynamics]: Competitive pressures between AI developers. {"
                - [Deception_By_AI]: AI systems deceiving humans about their true objectives
        - [Corrective_Feedback]: Human society implementing corrections after observing probl
            - [Warning_Shots]: Observable failures in weaker systems before catastrophic risk
            - [Rapid_Capability_Escalation]: AI capabilities escalating very rapidly, allowin
[Barriers_To_Understanding]: Difficulty in understanding the internal workings of advanced AI
- [Misaligned_Power_Seeking]: Deployed AI systems seeking power in unintended and high-impact
[Adversarial_Dynamics]: Potentially adversarial relationships between humans and power-seekin
- [Misaligned_Power_Seeking]: Deployed AI systems seeking power in unintended and high-impact
[Stakes_Of_Error]: The escalating impact of mistakes with power-seeking AI systems. {"instant
- [Misaligned_Power_Seeking]: Deployed AI systems seeking power in unintended and high-impact
```

This representation captures the complex causal structure of Carlsmith's argument, with 21 variables organized in a multi-level hierarchy. The "Misaligned_Power_Seeking" node appears

multiple times, reflecting its role as a central concept that influences several other variables.

After processing this structure with the AMTAIR system, probability information was added to create a complete BayesDown representation. The following excerpt shows the probability information for a single node ("Deployment_Decisions"):

```
[Deployment_Decisions]: Decisions to deploy potentially misaligned AI systems. {
  "instantiations": ["deployment_decisions_DEPLOY", "deployment_decisions_WITHHOLD"],
  "priors": {
    "p(deployment_decisions_DEPLOY)": "0.70",
    "p(deployment_decisions_WITHHOLD)": "0.30"
  },
  "posteriors": {
    "p(deployment_decisions_DEPLOY|incentives_to_build_aps_STRONG, deception_by_ai_TRUE)": "
    "p(deployment_decisions_DEPLOY|incentives_to_build_aps_STRONG, deception_by_ai_FALSE)": "
    "p(deployment_decisions_DEPLOY|incentives_to_build_aps_WEAK, deception_by_ai_TRUE)": "0.
    "p(deployment_decisions_DEPLOY|incentives_to_build_aps_WEAK, deception_by_ai_FALSE)": "0
  }
}
```

This node has two possible states (DEPLOY or WITHHOLD), prior probabilities for each state, and conditional probabilities based on different combinations of its parent variables ("Incentives_To_Build_APS" and "Deception_By_AI").

The complete BayesDown representation was processed through the AMTAIR pipeline, resulting in a structured DataFrame and ultimately a Bayesian network. Key extraction steps included:

```
# Extract structured data from BayesDown
carlsmith_df = parse_markdown_hierarchy_fixed(carlsmith_bayesdown, ArgDown=False)

# Enhance with calculated metrics
enhanced_carlsmith_df = enhance_extracted_data(carlsmith_df)

# Create network and visualization
carlsmith_network = create_bayesian_network(enhanced_carlsmith_df)
carlsmith_visualization = create_bayesian_network_with_probabilities(enhanced_carlsmith_df)
```

The resulting visualization (Figure 10) shows the complete Carlsmith model with color-coded nodes representing probability values:

[FIGURE 10: Interactive visualization of Carlsmith's model showing color-coded nodes and relationships]

This visualization reveals several structural insights:

1. **Central importance of "Misaligned_Power_Seeking"** as a hub node with multiple parents and children
2. **Multiple pathways to "Existential_Catastrophe"** through different intermediate factors
3. **Clusters of related variables** forming coherent subarguments (e.g., factors affecting alignment difficulty)
4. **Flow of influence** from technical factors (bottom) through deployment decisions to ultimate outcomes (top)

The implementation successfully handles the complexity of Carlsmith's model, correctly processing the multi-level structure, resolving repeated node references, and calculating appropriate probability distributions. The interactive visualization makes this complex model accessible, allowing users to explore different aspects of the argument through intuitive navigation.

Several key aspects of the implementation were particularly important for handling this complex model:

1. The **parent-child relationship detection algorithm** correctly identified hierarchical relationships despite the complex structure with repeated nodes and multiple levels.

2. The **probability question generation system** created appropriate questions for all variables, including those with multiple parents requiring factorial combinations of conditional probabilities.

3. The **network enhancement functions** calculated useful metrics like centrality measures and Markov blankets that help interpret the model structure.

4. The **visualization system** effectively presented the complex network through color-coding, interactive exploration, and progressive disclosure of details.

The successful application to Carlsmith's model demonstrates the AMTAIR approach's scalability to complex real-world arguments. While the canonical rain-sprinkler-lawn example validated correctness, this application proves practical utility for sophisticated multi-level arguments with dozens of variables and complex interdependencies—precisely the kind of arguments that characterize AI risk assessments.

This capability addresses a core limitation of the original MTAIR framework: the labor intensity of manual formalization. Where manually converting Carlsmith's argument to a formal model might take days of expert time, the AMTAIR approach accomplished this in minutes, creating a foundation for further analysis and exploration.