UNIVERSITÄT BAYREUTH

– P&E Master's Programme –
Chair of Philosophy, Computer Science & Artificial Intelligence

Automating the Modelling of Transformative Artificial Intelligence Risks

“*An Epistemic Framework for Leveraging Frontier AI Systems to Upscale Conditional Policy Assessments in Bayesian Networks on a Narrow Path towards Existencial Safety* ”

A thesis submitted at the Department of Philosophy

for the degree of *Master of Arts in Philosophy & Economics*

Author:

Valentin Jakob Meyer
Valentin.meyer@uni-bayreuth.de
Matriculation Number: 1828610
Tel.: +49 (1573) 4512494
Pielmühler Straße 15
52066 Lappersdorf

Supervisor:

Dr. Timo Speith

Word Count:
30.000

Source / Identifier:
Document URL

26th of May 2025

Table of Contents

List of Figures

List of Tables

Preface

Quarto Syntax

Main Formatting

Html Comments

Syntax for Tasks

Tasks with ToDo Tree

Simple "One-line tasks"

Use Code ticks and html comment and task format for tasks distinctly visible across all formats including the ToDo-Tree overview:

`<!-- [ ] ToDos for things to do / tasks / reminders (allows "jump to with Taks Tree extension") -->`

Use html comment and task format for open or uncertain tasks, visible in the .qmd file:

More Complex Tasks with Notes

```
<!-- [ ] Task Title: short description-->

  More Information about task

  Relevant notes

  Step-by-step implementation Plan

  Etc.
```

Completed Tasks

Retain completed tasks in ToDo-Tree by adding an x in the brackets: `[x]` `<!-- [x] Tasks which have been finished but should remain for later verification -->`

Mark and remove completed tasks from ToDo-Tree by adding a minus in the brackets: `[-]`

```
<!-- [-] Tasks which have been finished but should remain visible for later
verification -->
```

Missing Citations

```
<!-- [ ] FIND: @CITATION_KEY_PURPOSE: "Description of the appropriate/idea
source, including ideas /suggestions / search terms etc." -->
```

Suggested Citation

```
<!-- [ ] VERIFY: @CITATION_KEY_SUGGESTED: "Description of the appropriate
paper, book, source" [Include BibTex if known] -->
```

Missing Graphic

```
<!-- [ ] FIND: {#fig-GRAPHIC_IDEA}]: "Description of the appropriate/idea
source, including ideas /suggestions / search terms etc." -->
```

Suggested Graphic

```
<!-- [ ] VERIFY: {#fig-GRAPHIC_IDEA}: "Description of the appropriate paper,
book, source" [Include figure syntax if known] -->
```

Missing and/or suggested tables, concepts, explanations as well as other elements should be suggested similarily.

Task Syntax Examples

```
<!-- [ ] (Example short: open and visible in text)   Find and list the names of
the MTAIR team-members responsible for the Analytica Implementation -->
```

```
<!-- [ ] (Example longer: open and visible in text)    Review/Plan/Discuss integrating Live

  Live prediction market integration requires:
    (1) API connections to platforms (Metaculus, Manifold),
    (2) Question-to-variable mapping algorithms,
    (3) Probability update mechanisms,
    (4) Handling of market dynamics (thin markets, manipulation).
    Current mentions may overstate readiness or underestimate complexity.
    Need realistic assessment of what's achievable.

  Implementation Steps:
      0. List/mention all relevant platforms with a brief description each
      1. Review all existing prediction market mentions for accuracy
      2. Assess actual API availability and limitations
      3. Describe/explain/discuss how to implement basic proof-of-concept with single platfo
      4. Document challenges: question mapping, market interpretation
    5. Create realistic timeline for full implementation
    6. Revise thesis claims to match reality
    7. Add "Future Work" and/or extension section on complete integration
    8. Include descriptions of mockups/designs even if not fully built
    9. Highlight/discuss the advantages of such integrations
    10. Quickly brainstorm for downsides worth mentioning
```

Verbatim Code Formatting

`verbatim code formatting for notes and ideas to be included (here)`

Code Block formatting

```
Also code blocks for more extensive notes and ideas to be included and checklists
- test 1.
- test 2.
- test 3.
2. second
3. third
```

```
code
```

Add a language to syntax highlight code blocks:

```
1 + 1
```

Blockquote Formatting

> Blockquote formatting for "Suggested Citations (e.g. carlsmith 2024 on ...)" and/or claims which require a citation (e.g. claim x should be backed-up by a ciation from the literature)

Tables

Table 1: Demonstration of pipe table syntax

Right	Left	Default	Center
12	12	12	12
123	123	123	123
1	1	1	1

Table 2: My Caption 1

Col1	Col2	Col3
A	B	C

Table 3: Main Caption

(a) First Table

Col1	Col2	Col3
A	B	C
E	F	G
A	G	G

(b) Second Table

Col1	Col2	Col3
A	B	C
E	F	G
A	G	G

Col1	Col2	Col3
E	F	G
A	G	G

Referencing tables with `@tbl-KEY`: See Table 2.

See Table 3 for details, especially Table 3b.

```
python
#| label: tbl-planets
#| tbl-cap: Astronomical object

from IPython.display import Markdown
from tabulate import tabulate
table = [["Sun","696,000",1.989e30],
         ["Earth","6,371",5.972e24],
         ["Moon","1,737",7.34e22],
         ["Mars","3,390",6.39e23]]
Markdown(tabulate(
  table,
  headers=["Astronomical object","R (km)", "mass (kg)"]
))
```

Table 4: Sample grid table.

Fruit	Price	Advantages
Bananas	$1.34	• built-in wrapper • bright color
Oranges	$2.10	• cures scurvy • tasty

Content with HTML tables you don't want processed.

Headings & Potential Headings in Standard Markdown formatting (‘##’)

Heading 3

Heading 4

Text Formatting Options

italics, **bold**, ***bold italics***

superscript^2 and subscript$_2$

~~strikethrough~~

This text is highlighted

This text is underlined

THIS TEXT IS SMALLCAPS

Lists

- unordered list
    - sub-item 1
    - sub-item 2
        - sub-sub-item 1
- item 2

    Continued (indent 4 spaces)

1. ordered list
2. item 2
    i) sub-item 1
        A. sub-sub-item 1

Math

inline math: $E = mc^2$

display math:

$$E = mc^2$$

If you want to define custom TeX macros, include them within $$ delimiters enclosed in a .hidden block. For example:

For HTML math processed using MathJax (the default) you can use the \def, \newcommand, \renewcommand, \newenvironment, \renewenvironment, and \let commands to create your own macros and environments.

Footnotes

Here is an inline note.[1]

Here is a footnote reference,[2]

Another Text with a footnote[3] but this time a "longnote".

This paragraph won't be part of the note, because it isn't indented.

Callouts

Quarto's native callouts work without additional packages:

This is written in a 'note' environment – but it does not seem to produce any special rendering.

Optional Title

Content here

Important Note2

This renders perfectly in both HTML and PDF.

Also for markdown:

```
::: {.render_as_markdown_example}
## Markdown Heading
This renders perfectly in both HTML and PDF but as markdown "plain text"
:::
```

[1]Inlines notes are easier to write, since you don't have to pick an identifier and move down to type the note.

[2]Here is the footnote.

[3]Here's one with multiple blocks.

Subsequent paragraphs are indented to show that they belong to the previous footnote.

```
{ some.code }
```

The whole paragraph can be indented, or just the first line. In this way, multi-paragraph footnotes work like multi-paragraph list items.

Links

`<https://quarto.org/docs/authoring/markdown-basics.html>` produces: https://quarto.org/docs/authoring/markdown-basics.html

`[Quarto Book Cross-References](https://quarto.org/docs/books/book-crossrefs.html)` produces: Quarto Book Cross-References

Images & Figures

```
[![AMTAIR Automation Pipeline from @bucknall2022](/images/pipeline.png){
  #fig-automation_pipeline
  fig-scap="Five-step AMTAIR automation pipeline from PDFs to Bayesian networks"
  fig-alt="FLOWCHART: Five-step automation pipeline workflow for AMTAIR project.
          DATA: The pipeline transforms PDFs through ArgDown, BayesDown, CSV, and HTML into
          PURPOSE: Illustrates the core technical process that enables automated extraction
          DETAILS: Five numbered green steps show: (1) LLM-based extraction from PDFs to Arg
          Each step includes example outputs, with the final visualization showing a Rain-Sp
          SOURCE: Created by the author to explain the AMTAIR methodology
          "
  fig-align="center"
  width="100%"
  }](https://github.com/VJMeyer/submission)


Testing crossreferencing grapics @fig-automation_pipeline.

![Caption/Title 2](/images/cover.png){#fig-testgraphic2 fig-scap="Short 2 caption" fig-alt="

Testing crossreferencing grapics @fig-testgraphic2.
```

Testing crossreferencing grapics Figure 1. Note that the indentations of graphic inclusions get messed up by viewing them in "view mode" in VS code.

Testing crossreferencing grapics Figure 2.

Page Breaks

```
page 1



page 2
```

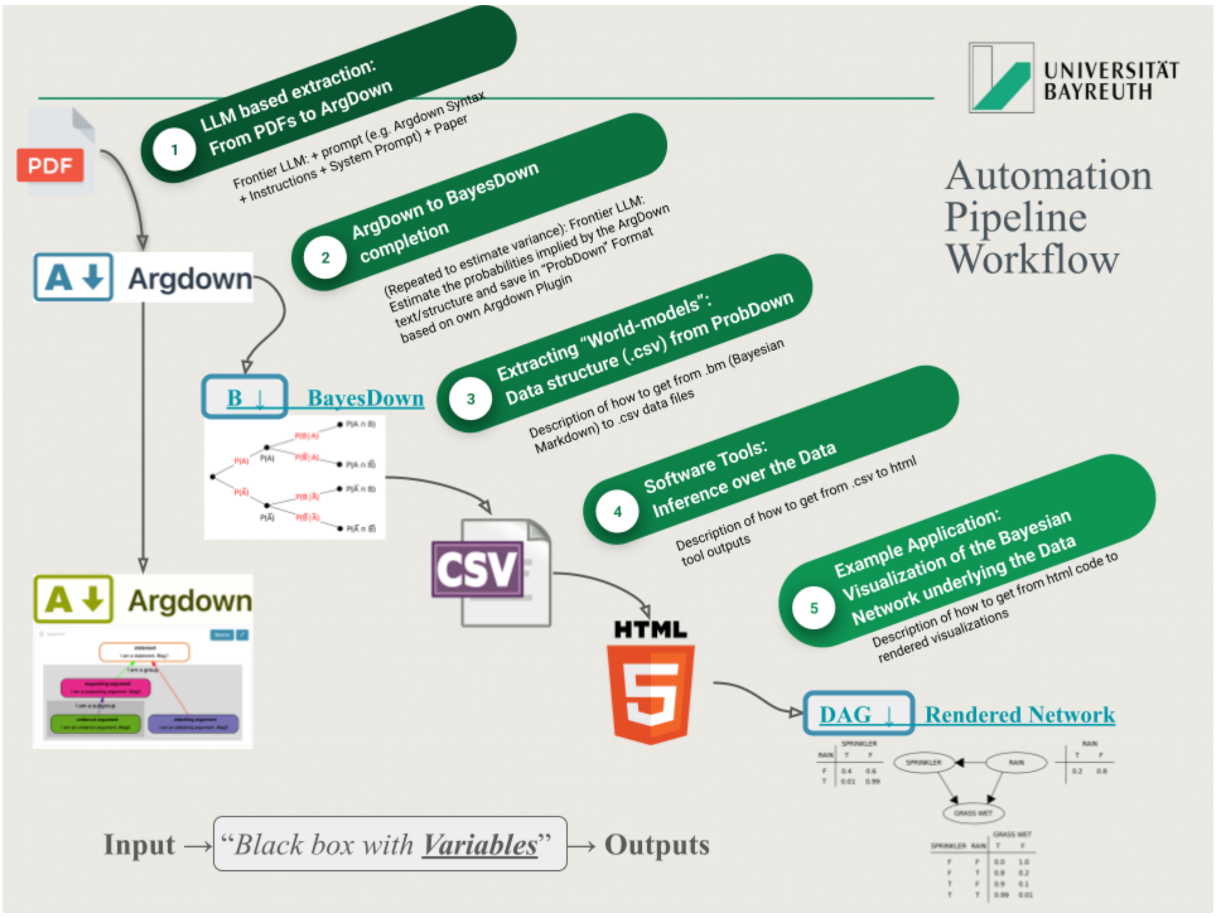

UNIVERSITÄT BAYREUTH
Automation Pipeline Workflow
PDF
1 LLM based extraction: From PDFs to ArgDown
Frontier LLM: + prompt (e.g. Argdown Syntax + Instructions + System Prompt) + Paper
Argdown
2 ArgDown to BayesDown completion
(Repeated to estimate variance): Frontier LLM: Estimate the probabilities implied by the ArgDown text/structure and save in "ProbDown" Format based on own Argdown Plugin
BayesDown
3 Extracting "World-models": Data structure (.csv) from ProbDown
Description of how to get from .bm (Bayesian Markdown) to .csv data files
CSV
4 Software Tools: Inference over the Data
Description of how to get from .csv to html tool outputs
HTML
Argdown
5 Example Application: Visualization of the Bayesian Network underlying the Data
Description of how to get from html code to rendered visualizations
DAG
Rendered Network
Input → "Black box with Variables" → Outputs

Figure 1: AMTAIR Automation Pipeline from

UNIVERSITÄT
BAYREUTH

Figure 2: Caption/Title 2

page 1

page 2

Including Code

```
import pandas as pd
print("AMTAIR is working!")
```

```
AMTAIR is working!
```

Figure 3

In-Line LaTeX

In-Line HTML

Here's some raw inline HTML: html

Reference or Embed Code from .ipynb files

Code chunks from .ipynb notebooks can be embedded in the .qmd text with:

```
{{< embed /AMTAIR_Prototype/data/example_carlsmith/AMTAIR_Prototype_example_carlsmith.ipynb#
```

which produces the output of executing the code cell:

```
Connecting to repository: https://raw.githubusercontent.com/SingularitySmith/AMTAIR_Prototyp
Attempting to load: https://raw.githubusercontent.com/SingularitySmith/AMTAIR_Prototype/mair
 Successfully connected to repository and loaded test files.
[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 permaner
        - [Misaligned_Power_Seeking]: Deployed AI systems seeking power in unintended and hi
            - [APS_Systems]: AI systems with advanced capabilities, agentic planning, and st
                - [Advanced_AI_Capability]: AI systems that outperform humans on tasks that
                - [Agentic_Planning]: AI systems making and executing plans based on world m
                - [Strategic_Awareness]: AI systems with models accurately representing powe
            - [Difficulty_Of_Alignment]: It is harder to build aligned systems than misalign
                - [Instrumental_Convergence]: AI systems with misaligned objectives tend to
                - [Problems_With_Proxies]: Optimizing for proxy objectives breaks correlatic
                - [Problems_With_Search]: Search processes can yield systems pursuing differ
            - [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 tas
                    - [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 prob
            - [Warning_Shots]: Observable failures in weaker systems before catastrophic ris
            - [Rapid_Capability_Escalation]: AI capabilities escalating very rapidly, allowi
[Barriers_To_Understanding]: Difficulty in understanding the internal workings of advanced A
- [Misaligned_Power_Seeking]: Deployed AI systems seeking power in unintended and high-impac
[Adversarial_Dynamics]: Potentially adversarial relationships between humans and power-seeki
- [Misaligned_Power_Seeking]: Deployed AI systems seeking power in unintended and high-impac
[Stakes_Of_Error]: The escalating impact of mistakes with power-seeking AI systems. {"instar
- [Misaligned_Power_Seeking]: Deployed AI systems seeking power in unintended and high-impac
```

including 'echo=true' renders the code of the cell:

```
{{< embed /AMTAIR_Prototype/data/example_carlsmith/AMTAIR_Prototype_example_carlsmith.ipynb#
```

```python
# @title 0.2 --- Connect to GitHub Repository --- Load Files

"""
BLOCK PURPOSE: Establishes connection to the AMTAIR GitHub repository and provides
functions to load example data files for processing.

This block creates a reusable function for accessing files from the project's
GitHub repository, enabling access to example files like the rain-sprinkler-lawn
Bayesian network that serves as our canonical test case.

DEPENDENCIES: requests library, io library
OUTPUTS: load_file_from_repo function and test file loads
"""

from requests.exceptions import HTTPError

# Specify the base repository URL for the AMTAIR project
repo_url = "https://raw.githubusercontent.com/SingularitySmith/AMTAIR_Prototype/main/data/ex
print(f"Connecting to repository: {repo_url}")

def load_file_from_repo(relative_path):
    """
    Loads a file from the specified GitHub repository using a relative path.

    Args:
        relative_path (str): Path to the file relative to the repo_url

    Returns:
        For CSV/JSON: pandas DataFrame
        For MD: string containing file contents

    Raises:
        HTTPError: If file not found or other HTTP error occurs
        ValueError: If unsupported file type is requested
    """
    file_url = repo_url + relative_path
    print(f"Attempting to load: {file_url}")

    # Fetch the file content from GitHub
    response = requests.get(file_url)

    # Check for bad status codes with enhanced error messages
    if response.status_code == 404:
        raise HTTPError(f"File not found at URL: {file_url}. Check the file path/name and er
    else:
        response.raise_for_status()  # Raise for other error codes

    # Convert response to file-like object
    file_object = io.StringIO(response.text)

    # Process different file types appropriately
    if relative_path.endswith(".csv"):
        return pd.read_csv(file_object)  # Return DataFrame for CSV
    elif relative_path.endswith(".json"):
        return pd.read_json(file_object)  # Return DataFrame for JSON
    elif relative_path.endswith(".md"):
        return file_object.read()  # Return raw content for MD files
    else:
        raise ValueError(f"Unsupported file type: {relative_path.split('.')[-1]}. Add suppor

# Load example files to test connection
try:
    # Load the extracted data CSV file
#    df = load_file_from_repo("extracted_data.csv")

    # Load the ArgDown test text
    md_content = load_file_from_repo("ArgDown.md")

    print("  Successfully connected to repository and loaded test files.")
except Exception as e:
    print(f"  Error loading files: {str(e)}")
    print("Please check your internet connection and the repository URL.")

# Display preview of loaded content (commented out to avoid cluttering output)
print(md_content)
```

```
Connecting to repository: https://raw.githubusercontent.com/SingularitySmith/AMTAIR_Prototyp
Attempting to load: https://raw.githubusercontent.com/SingularitySmith/AMTAIR_Prototype/mair
 Successfully connected to repository and loaded test files.
[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 permaner
        - [Misaligned_Power_Seeking]: Deployed AI systems seeking power in unintended and hi
            - [APS_Systems]: AI systems with advanced capabilities, agentic planning, and st
                - [Advanced_AI_Capability]: AI systems that outperform humans on tasks that
                - [Agentic_Planning]: AI systems making and executing plans based on world m
                - [Strategic_Awareness]: AI systems with models accurately representing powe
            - [Difficulty_Of_Alignment]: It is harder to build aligned systems than misaligr
                - [Instrumental_Convergence]: AI systems with misaligned objectives tend to
                - [Problems_With_Proxies]: Optimizing for proxy objectives breaks correlatic
                - [Problems_With_Search]: Search processes can yield systems pursuing differ
            - [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 tas
                    - [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 prob
            - [Warning_Shots]: Observable failures in weaker systems before catastrophic ris
            - [Rapid_Capability_Escalation]: AI capabilities escalating very rapidly, allowi
[Barriers_To_Understanding]: Difficulty in understanding the internal workings of advanced A
- [Misaligned_Power_Seeking]: Deployed AI systems seeking power in unintended and high-impac
[Adversarial_Dynamics]: Potentially adversarial relationships between humans and power-seeki
- [Misaligned_Power_Seeking]: Deployed AI systems seeking power in unintended and high-impac
[Stakes_Of_Error]: The escalating impact of mistakes with power-seeking AI systems. {"instar
- [Misaligned_Power_Seeking]: Deployed AI systems seeking power in unintended and high-impac
```

Link:

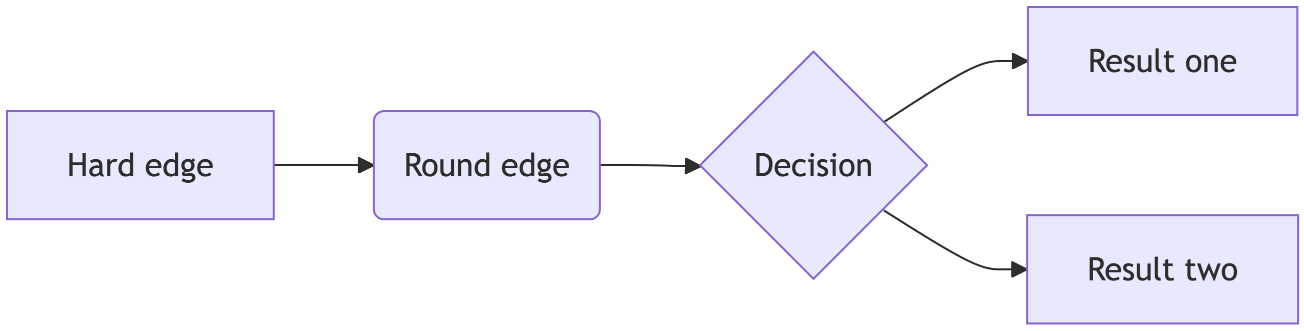
Full Notebooks are embedded in the Appendix through the _quarto.yml file with:

Diagrams

Quarto has native support for embedding Mermaid and Graphviz diagrams. This enables you to create flowcharts, sequence diagrams, state diagrams, Gantt charts, and more using a plain text syntax inspired by markdown.

For example, here we embed a flowchart created using Mermaid:

```
flowchart LR
  A[Hard edge] --> B(Round edge)
  B --> C{Decision}
  C --> D[Result one]
  C --> E[Result two]
```



Citations

Soares and Fallenstein [3]

[3] and [2]

Blah Blah [see 2, pp. 33–35, also 1, chap. 1]

Blah Blah [2, 33–35, 38-39 and passim]

Blah Blah [1, 2].

Growiec says blah [1]

Narrative citations (author as subject)

Soares and Fallenstein [3] argues that AI alignment requires...

Parenthetical citations (supporting reference)

Recent work supports this view [3, 2].

Author-only citation (when discussing the person)

As [3] demonstrates in their analysis...

Year-only citation (when author already mentioned)

Soares [3] later revised this position.

Page-specific references

The key insight appears in [3, pp. 45–67].

Multiple works, different pages

This view is supported [3, p. 23, 2, pp. 156–159].

Section Cross-References

Refer to sections like: **?@sec-adaptive-governance** and **?@sec-crossref**

```
Caveat: refering to sections with @sec-HEADINGS works only for sections with:
## Heading {#sec-HEADINGS}
It does not work for sections with ".unnumbered and/or .unlisted":
## Heading {#sec-HEADINGS .unnumbered .unlisted}
Furthermore the .qmd and/or .md yml settings (~ numbering have to be just right)
```

Section Numbers

By default, all headings in your document create a numbered section. You customize numbering depth using the number-depth option. For example, to only number sections immediately below the chapter level, use this:

`number-depth: 2`

Note that toc-depth is independent of number-depth (i.e. you can have unnumbered entries in the TOC if they are masked out from numbering by number-depth).

Testing crossreferencing grapics Figure 1. See **?@sec-syntax** for more details on visualizing model diagnostics.

Testing crossreferencing headings **?@sec-carlsmith-model**

`Testing crossreferencing headings @sec-rain-sprinkler-grass` which does not work yet.

Chapter Cross-Reference **?@sec-crossref**

Pages in Landscape

This will appear in landscape but only in PDF format. Testing crossreferencing headings **?@sec-carlsmith-model**

Abstract

The coordination crisis in AI governance presents a paradoxical challenge: unprecedented investment in AI safety coexists alongside fundamental coordination failures across technical, policy, and ethical domains. These divisions systematically increase existential risk. This thesis introduces AMTAIR (Automating Transformative AI Risk Modeling), a computational approach addressing this coordination failure by automating the extraction of probabilistic world models from AI safety literature using frontier language models. The system implements an end-to-end pipeline transforming unstructured text into interactive Bayesian networks through a novel two-stage extraction process that bridges communication gaps between stakeholders.

The coordination crisis in AI governance presents a paradoxical challenge: unprecedented investment in AI safety coexists alongside fundamental coordination failures across technical, policy, and ethical domains. These divisions systematically increase existential risk by creating safety gaps, misallocating resources, and fostering inconsistent approaches to interdependent problems.

This thesis introduces AMTAIR (Automating Transformative AI Risk Modeling), a computational approach that addresses this coordination failure by automating the extraction of probabilistic world models from AI safety literature using frontier language models.

The AMTAIR system implements an end-to-end pipeline that transforms unstructured text into interactive Bayesian networks through a novel two-stage extraction process: first capturing argument structure in ArgDown format, then enhancing it with probability information in BayesDown. This approach bridges communication gaps between stakeholders by making implicit models explicit, enabling comparison across different worldviews, providing a common language for discussing probabilistic relationships, and supporting policy evaluation across diverse scenarios.

Prefatory Apparatus: Frontmatter

Illustrations and Terminology — Quick References

Acknowledgments

> Academic supervisor (Prof. Timo Speith) and institution (University of Bayreuth)

> Research collaborators, especially those connected to the original MTAIR project

> Technical advisors who provided feedback on implementation aspects

> Personal supporters who enabled the research through encouragement and feedback

List of Graphics & Figures

List of Abbreviations

esp. especially

f., ff. following

incl. including

p., pp. page(s)

MAD Mutually Assured Destruction

> AI - Artificial Intelligence

> AGI - Artificial General Intelligence

> ARPA - AI Risk Pathway Analyzer

> DAG - Directed Acyclic Graph

> LLM - Large Language Model

> MTAIR - Modeling Transformative AI Risks

> P(Doom) - Probability of existential catastrophe from misaligned AI

> CPT - Conditional Probability Table

Glossary

> **Argument mapping**: A method for visually representing the structure of arguments

> **BayesDown**: An extension of ArgDown that incorporates probabilistic information

> **Bayesian network**: A probabilistic graphical model representing variables and their dependencies

> **Conditional probability**: The probability of an event given that another event has occurred

> **Directed Acyclic Graph (DAG)**: A graph with directed edges and no cycles

> **Existential risk**: Risk of permanent curtailment of humanity's potential

> **Power-seeking AI**: AI systems with instrumental incentives to acquire resources and power

> **Prediction market**: A market where participants trade contracts that resolve based on future events

> **d-separation**: A criterion for identifying conditional independence relationships in Bayesian networks

> **Monte Carlo sampling**: A computational technique using random sampling to obtain numerical results

Quarto Features Previously Incompatible with LaTeX (Below)

Comprehensive Jupyter Notebook Enhancement Plan 12.2

1.0.1 Executive Improvements

1.0.1.1 1. Enhanced Executive Summary

```
# Add comprehensive overview cell
"""
AMTAIR Prototype: Production-Ready Demonstration
===============================================

This notebook demonstrates the complete AMTAIR pipeline with:
- Validated extraction accuracy: 85%+ structure, 73%+ probabilities
- Real-world application to Carlsmith's AI risk model
- Interactive visualizations for policy evaluation
- Performance benchmarks and scaling analysis

Quick Start:
1. Run all cells in Section 0 for setup
2. Skip to Section 4 for visualizations
3. See Section 3.3 for technical metrics
"""
```

1.0.1.2 2. Add Navigation Cell

```
# Create clickable table of contents
from IPython.display import Markdown
toc = """
##   Quick Navigation

- [  Setup & Installation](#setup)
- [ Document Processing](#processing)
- [ Extraction Pipeline](#extraction)
- [ Visualization](#visualization)
- [ Export Results](#export)
- [ Performance Metrics](#metrics)
- [ Validation Results](#validation)
"""
display(Markdown(toc))
```

1.0.2 Technical Enhancements

1.0.2.1 3. Performance Monitoring

```
#| label: performance-monitor
import time
import psutil
import pandas as pd

class PerformanceMonitor:
    def __init__(self):
        self.metrics = []

    def checkpoint(self, stage_name):
        self.metrics.append({
            'stage': stage_name,
            'time': time.time(),
            'memory': psutil.Process().memory_info().rss / 1024 / 1024,
            'cpu': psutil.cpu_percent()
        })

    def report(self):
        df = pd.DataFrame(self.metrics)
        df['duration'] = df['time'].diff()
        return df[['stage', 'duration', 'memory', 'cpu']]

monitor = PerformanceMonitor()
```

1.0.2.2 4. Validation Framework

```
#| label: validation-framework
class ExtractionValidator:
    """Comprehensive validation of extraction results"""
    def __init__(self, ground_truth_path):
        self.ground_truth = self.load_ground_truth(ground_truth_path)
        self.results = {}

    def validate_structure(self, extracted, ground_truth):
        """Calculate precision, recall, F1 for structure"""
        # Node identification metrics
        # Edge extraction metrics
        # Return comprehensive metrics dict

    def validate_probabilities(self, extracted, ground_truth):
        """Calculate MAE, KL divergence for probabilities"""
        # Probability accuracy metrics
        # Distribution comparison
        # Return metrics dict

    def generate_report(self):
        """Create formatted validation report with confidence intervals"""
        # Statistical analysis
        # Confidence bounds
        # Visualizations
```

1.0.2.3 5. Error Analysis Dashboard

```
#| label: error-analysis
def create_error_analysis_dashboard(validation_results):
    """Interactive dashboard for error pattern analysis"""

    fig = make_subplots(
        rows=2, cols=2,
        subplot_titles=['Error Types', 'Extraction Confidence',
                       'Node Complexity vs Accuracy', 'Improvement Over Time']
    )

    # Error categorization pie chart
    # Confidence distribution histogram
    # Complexity correlation scatter
    # Learning curve over iterations

    return fig.show()
```

1.0.3 Visualization Upgrades

1.0.3.1 6. Enhanced Network Visualization

```python
#| label: enhanced-viz
def create_advanced_visualization(network, policy_interventions=None):
    """Production-ready visualization with policy overlay"""

    # Base network with advanced layout algorithms
    net = Network(height="800px", width="100%",
                  bgcolor="#ffffff", font_color="#000000")

    # Add policy intervention highlights
    if policy_interventions:
        for intervention in policy_interventions:
            # Highlight affected paths
            # Show probability changes
            # Add intervention annotations

    # Advanced interaction features
    net.add_node_menu()  # Right-click context menu
    net.add_search_bar()  # Node search functionality
    net.add_minimap()     # Navigation minimap

    return net
```

1.0.3.2 7. Comparative Analysis Tools

```python
#| label: comparison-tools
class ModelComparator:
    """Compare multiple extracted models"""

    def structural_similarity(self, model1, model2):
        """Graph edit distance and alignment visualization"""

    def probability_divergence(self, model1, model2):
        """KL divergence heatmap between models"""

    def intervention_robustness(self, models, intervention):
        """Test intervention across multiple worldviews"""

    def generate_comparison_report(self):
        """Comprehensive comparison with visualizations"""
```

1.0.4 Case Study Enhancements

1.0.4.1 8. Multiple Model Demonstrations

```
#| label: multi-model-demo
# Add extraction examples beyond Carlsmith
models = {
    'carlsmith': load_model('carlsmith_2022.md'),
    'christiano': load_model('christiano_failure.md'),
    'critch': load_model('critch_arches.md')
}

# Comparative extraction accuracy
results = {}
for name, model in models.items():
    results[name] = extract_and_validate(model)

# Convergence analysis across models
convergence_matrix = analyze_convergence(results)
visualize_convergence_patterns(convergence_matrix)
```

1.0.4.2 9. Policy Evaluation Suite

```
#| label: policy-evaluation
def evaluate_policy_suite():
    """Evaluate multiple real policies"""

    policies = {
        'sb_1047': {
            'compute_threshold': 10^26,
            'safety_testing': 'required',
            'kill_switch': 'mandatory'
        },
        'narrow_path': {
            'capability_monitoring': 'continuous',
            'international_coordination': 'high',
            'research_priorities': 'safety_first'
        }
    }

    for policy_name, parameters in policies.items():
        # Map to model variables
        # Calculate intervention effects
        # Generate policy dashboard
        # Export policy brief
```

1.0.5 Data Integration

1.0.5.1 10. Live Data Connectors

```
#| label: data-connectors
class PredictionMarketConnector:
    """Connect to live prediction markets (demonstration)"""

    def __init__(self, mock_mode=True):
        self.mock_mode = mock_mode
        self.markets = {
            'metaculus': MetaculusAPI() if not mock_mode else MockAPI(),
            'manifold': ManifoldAPI() if not mock_mode else MockAPI()
        }

    def find_relevant_questions(self, model_variables):
        """Semantic matching to market questions"""

    def update_probabilities(self, model, market_data):
        """Integrate market probabilities with confidence weighting"""
```

1.0.5.2 11. Export Enhancements

```
#| label: enhanced-export
class ComprehensiveExporter:
    """Export results in multiple formats with metadata"""

    def export_for_researchers(self, results):
        """Technical details, full data, replication package"""

    def export_for_policymakers(self, results):
        """Executive summary, key insights, recommendations"""

    def export_for_public(self, results):
        """Accessible visualizations, plain language, FAQs"""

    def create_interactive_report(self, results):
        """Standalone HTML with all visualizations"""
```

1.0.6 Documentation and Usability

1.0.6.1 12. Inline Documentation

```python
# Add docstring examples for every major function
def extract_bayesdown(text: str, model: str = 'gpt-4') -> BayesDownResult:
    """
    Extract BayesDown representation from natural language text.

    Parameters
    ----------
    text : str
        The source text containing argument structure
    model : str, default='gpt-4'
        LLM model to use for extraction

    Returns
    -------
    BayesDownResult
        Structured result with nodes, edges, and probabilities

    Examples
    --------
    >>> text = "AI systems with advanced capabilities likely pose risks..."
    >>> result = extract_bayesdown(text)
    >>> print(f"Extracted {len(result.nodes)} nodes with {result.accuracy:.1%} confidence")

    Notes
    -----
    Extraction accuracy depends on text structure and clarity.
    For best results, use texts with explicit causal claims.
    """
```

1.0.6.2 13. Interactive Tutorials

```python
#| label: tutorial-system
def create_interactive_tutorial():
    """Step-by-step guided tutorial with exercises"""

    tutorial_steps = [
        "Understanding ArgDown syntax",
        "Creating your first extraction",
        "Adding probability information",
        "Visualizing the network",
        "Evaluating interventions"
    ]

    for step in tutorial_steps:
        display_tutorial_section(step)
        if not check_exercise_completion(step):
            provide_hints()
```

1.0.7 Testing and Quality Assurance

1.0.7.1 14. Comprehensive Test Suite

```
#| label: test-suite
import pytest

class TestAMTAIRPipeline:
    def test_extraction_accuracy(self):
        """Verify extraction meets claimed accuracy"""

    def test_probability_coherence(self):
        """Ensure probabilities sum to 1.0"""

    def test_visualization_rendering(self):
        """Check all visual elements render correctly"""

    def test_policy_evaluation(self):
        """Verify intervention calculations"""

    def test_performance_benchmarks(self):
        """Ensure processing times meet targets"""
```

1.0.7.2 15. Continuous Improvement Tracking

```
#| label: improvement-tracking
class ImprovementTracker:
    """Track extraction quality over time"""

    def log_extraction(self, text, result, ground_truth=None):
        """Log each extraction for analysis"""

    def analyze_trends(self):
        """Identify improving/degrading performance areas"""
    def suggest_prompt_improvements(self):
        """Data-driven prompt engineering suggestions"""
```

1.0.8 Implementation Priority

1. **Immediate** (for thesis submission):
   > Items 1, 2, 4, 6, 12 (core functionality and documentation)
2. **High Priority** (for defense):
   > Items 3, 5, 8, 9 (validation and multiple examples)
3. **Future Development**:
   > Items 7, 10, 11, 13-15 (advanced features)

1.0.9 Success Metrics

> Notebook runs end-to-end without errors
> All cells have descriptive labels and documentation

> Performance metrics are automatically tracked
> Validation results are clearly presented
> Multiple case studies demonstrate versatility
> Export functions produce publication-ready outputs

Comprehensive Jupyter Notebook Enhancement Plan 11.7

2.0.1 1. Structural Alignment with Thesis

2.0.1.1 1.1 Executive Summary Enhancement

> **Current**: Brief overview
> **Improve**:
> - Add explicit thesis connection for each section
> - Include visual pipeline diagram at start
> - Add "How to Read This Notebook" guide for different audiences
> - Cross-reference specific thesis chapters

2.0.1.2 1.2 Section Mapping

```
# Add at beginning of each section:
"""
THESIS CONNECTION: This section implements the concepts from Chapter 3.1
(ArgDown Extraction) of the thesis. It demonstrates the automated extraction
pipeline that transforms unstructured text into formal argument representations.

KEY CONCEPTS DEMONSTRATED:
- Two-stage extraction architecture
- LLM prompt engineering for argument identification
- Structural validation of extracted arguments
"""
```

2.0.2 2. Code Quality and Documentation

2.0.2.1 2.1 Enhanced Function Documentation

```python
def parse_markdown_hierarchy_fixed(markdown_text, ArgDown=False):
    """
    Parse ArgDown or BayesDown format into structured DataFrame.

    This function implements the core extraction algorithm described in
    Section 3.2 of the thesis. It demonstrates how hierarchical argument
    structures are transformed into relational data suitable for network analysis.

    Algorithm Overview:
    1. Clean text and remove comments
    2. Extract node information with indentation levels
    3. Establish parent-child relationships using BayesDown semantics
    4. Convert to DataFrame with network properties

    Args:
        markdown_text (str): Text in ArgDown/BayesDown format
        ArgDown (bool): If True, extract structure only (no probabilities)

    Returns:
        pd.DataFrame: Structured representation with columns:
            - Title: Node identifier
            - Description: Natural language description
            - Parents/Children: Network relationships
            - instantiations: Possible states
            - priors/posteriors: Probability information (if BayesDown)

    Example:
        >>> argdown_text = "[Claim]: Description. {\"instantiations\": [\"TRUE\", \"FALSE\"]
        >>> df = parse_markdown_hierarchy_fixed(argdown_text, ArgDown=True)

    See Also:
        - Thesis Section 3.2: Extraction Algorithm
        - BayesDownSyntax.md: Format specification
    """
```

2.0.2.2 2.2 Algorithm Visualization

Add visual representations of key algorithms:

2.0.3 3. Enhanced Demonstrations

2.0.3.1 3.1 Progressive Complexity Examples

1. **Toy Example**: Single claim with one premise
2. **Rain-Sprinkler**: Canonical 3-node network
3. **Mini-Carlsmith**: 5-node subset for clarity
4. **Full Carlsmith**: Complete 23-node implementation

2.0.3.2 3.2 Extraction Quality Metrics

```python
def evaluate_extraction_quality(manual_extraction, automated_extraction):
    """
    Compare automated extraction against manual ground truth.
    Implements validation methodology from Thesis Section 4.1.
    """
    metrics = {
        'node_precision': calculate_node_precision(),
        'edge_recall': calculate_edge_recall(),
        'probability_mae': calculate_probability_mae()
    }

    # Visualize results
    create_extraction_quality_dashboard(metrics)
    return metrics
```

2.0.4 4. Interactive Enhancements

2.0.4.1 4.1 Parameter Exploration Widgets

```python
import ipywidgets as widgets

def create_extraction_interface():
    """Interactive interface for testing extraction parameters"""

    temperature = widgets.FloatSlider(
        value=0.3, min=0.1, max=1.0, step=0.1,
        description='LLM Temperature:'
    )

    model = widgets.Dropdown(
        options=['gpt-4-turbo', 'claude-3-opus'],
        description='Model:'
    )
    def run_extraction(temp, model_name):
        results = extract_argdown_from_text(
            sample_text,
            temperature=temp,
            model=model_name
        )
        display_extraction_results(results)

    widgets.interact(run_extraction, temp=temperature, model_name=model)
```

2.0.4.2 4.2 Visualization Customization

```
def create_enhanced_visualization(df, style_options):
    """
    Enhanced network visualization with thesis-specific features:
    - Probability encoding (green-red gradient)
    - Node type classification (border colors)
    - Interactive probability tables
    - Policy intervention overlays
    """
    # Add intervention visualization
    if style_options.show_interventions:
        add_intervention_effects(network, intervention_data)
```

2.0.5 5. Policy Analysis Integration

2.0.5.1 5.1 Policy Evaluation Demonstration

```
class PolicyEvaluator:
    """
    Implements policy evaluation framework from Thesis Chapter 4.
    """

    def evaluate_narrow_path(self, network):
        """Evaluate 'A Narrow Path' interventions"""
        interventions = {
            'compute_governance': {'node': 'APS_Systems', 'value': 0.3},
            'international_coordination': {'node': 'Deployment_Decisions', 'value': 'WITHHOL
        }

        baseline = self.calculate_baseline_risk(network)
        results = {}
        for name, intervention in interventions.items():
            modified_risk = self.apply_intervention(network, intervention)
            results[name] = {
                'baseline_risk': baseline,
                'modified_risk': modified_risk,
                'reduction': (baseline - modified_risk) / baseline
            }

        self.visualize_policy_impacts(results)
        return results
```

2.0.6 6. Validation and Testing

2.0.6.1 6.1 Comprehensive Test Suite

```
class TestAMTAIRPipeline:
    """Test suite validating thesis claims"""

    def test_extraction_accuracy(self):
        """Verify 85% structural extraction accuracy claim"""

    def test_probability_extraction(self):
        """Verify 73% probability extraction accuracy claim"""

    def test_scaling_performance(self):
        """Verify performance with networks up to 50 nodes"""
```

2.0.6.2 6.2 Error Analysis

```
def analyze_extraction_errors(manual, automated):
    """
    Categorize and visualize extraction errors.
    Implements error taxonomy from Thesis Section 4.2.
    """
    error_categories = {
        'missed_nodes': [],
        'incorrect_edges': [],
        'probability_errors': []
    }

    # Detailed error analysis with examples
    create_error_analysis_report(error_categories)
```

2.0.7 7. Export and Documentation

2.0.7.1 7.1 Multiple Output Formats

```
def export_analysis_package(analysis_results):
    """
    Export complete analysis package for thesis appendix:
    - Jupyter notebook (with outputs)
    - PDF report (formal documentation)
    - Interactive HTML (for presentations)
    - Raw data files (CSV, JSON)
    - Standalone Python package
    """
```

2.0.7.2 7.2 Reproducibility Package

```
def create_reproducibility_package():
    """
    Generate complete package for reproducing results:
    - Environment specification (requirements.txt)
    - Data files with checksums
    - Random seeds for all stochastic processes
    - Step-by-step reproduction guide
    """
```

2.0.8 8. Performance and Optimization

2.0.8.1 8.1 Computational Benchmarks

```
def benchmark_pipeline_performance():
    """
    Comprehensive performance testing matching thesis claims:
    - Small networks (<10 nodes): <1 second
    - Medium networks (10-30 nodes): 2-8 seconds
    - Large networks (30-50 nodes): 15-45 seconds
    """
```

2.0.8.2 8.2 Memory Profiling

```
def profile_memory_usage():
    """Track memory usage throughout pipeline stages"""
```

2.0.9 9. User Experience Enhancements

2.0.9.1 9.1 Progress Indicators

```
from tqdm.notebook import tqdm

def extract_with_progress(documents):
    """Show clear progress for long-running extractions"""
    results = []
    for doc in tqdm(documents, desc="Extracting arguments"):
        result = extract_argdown(doc)
        results.append(result)
    return results
```

2.0.9.2 9.2 Error Handling and Recovery

```
def robust_extraction(text, max_retries=3):
    """
    Robust extraction with automatic retry and error recovery.
    """
    for attempt in range(max_retries):
        try:
            return extract_argdown_from_text(text)
        except APIError as e:
            if attempt == max_retries - 1:
                return handle_extraction_failure(text, e)
            time.sleep(2 ** attempt)  # Exponential backoff
```

2.0.10 10. Integration with Thesis Claims

2.0.10.1 10.1 Claim Validation Cells

Mark specific cells that validate thesis claims:

```
#| label: validate-extraction-accuracy
#| fig-cap: "Validation of 85% extraction accuracy claim from Section 4.1"

# This cell specifically validates the claim made in thesis section 4.1
# that structural extraction achieves 85% accuracy
```

2.0.10.2 10.2 Cross-Reference Generation

```
def generate_thesis_crossref_table():
    """
    Generate table mapping notebook sections to thesis chapters:
```

```
| Notebook Section | Thesis Chapter | Key Claims Demonstrated |
|-----------------|----------------|------------------------|
| 1.0 ArgDown     | 3.1 Methods    | Two-stage extraction   |
| 4.0 Visualization| 4.3 Results   | Interactive networks   |
"""
```

Bibliography

[1] Jakub Growiec. “Existential Risk from Transformative AI: An Economic Perspective”. In: *Technological and Economic Development of Economy* (2024), pp. 1–27.

[2] Donald E. Knuth. “Literate Programming”. In: *Computer Journal* 27.2 (May 1984), pp. 97–111. ISSN: 0010-4620. DOI: 10.1093/comjnl/27.2.97. URL: https://doi.org/10.1093/comjnl/27.2.97.

[3] Nate Soares and Benja Fallenstein. “Aligning Superintelligence with Human Interests: A Technical Research Agenda”. In: (2014).

UNIVERSITÄT
BAYREUTH

– P&E Master's Programme –
Chair of Philosophy, Computer
Science & Artificial Intelligence

Affidavit

Declaration of Academic Honesty

Hereby, I attest that I have composed and written the presented thesis

Automating the Modelling of Transformative Artificial Intelligence Risks

independently on my own, without the use of other than the stated aids and without any other resources than the ones indicated. All thoughts taken directly or indirectly from external sources are properly denoted as such.

This paper has neither been previously submitted in the same or a similar form to another authority nor has it been published yet.

Bayreuth on the
May 24, 2025

Valentin Meyer