

# Chain-of-Regulation on Modularized Queries

## LLM Hallucination Mitigation

DIPLOMARBEIT

zur Erlangung des akademischen Grades

**Diplom-Ingenieur**

im Rahmen des Studiums

**Software Engineering und Internet Computing**

eingereicht von

**Alexander Gube, BSc**

Matrikelnummer 12023988

an der Fakultät für Informatik

der Technischen Universität Wien

Betreuung: Em.O.Univ.Prof. Dipl.-Ing. Dr.techn. Georg Gottlob

Mitwirkung: Univ.Lektor Dipl.-Ing. Bernhard Krüpl-Sypien

Wien, 13. Oktober 2025

---

Alexander Gube

---

Georg Gottlob



Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar  
The approved original version of this thesis is available in print at TU Wien Bibliothek.

# Chain-of-Regulation on Modularized Queries

## LLM Hallucination Mitigation

DIPLOMA THESIS

submitted in partial fulfillment of the requirements for the degree of

**Diplom-Ingenieur**

in

**Software Engineering and Internet Computing**

by

**Alexander Gube, BSc**

Registration Number 12023988

to the Faculty of Informatics

at the TU Wien

Advisor: Em.O.Univ.Prof. Dipl.-Ing. Dr.techn. Georg Gottlob

Assistance: Univ.Lektor Dipl.-Ing. Bernhard Krüpl-Sypien

Vienna, October 13, 2025

---

Alexander Gube

---

Georg Gottlob



Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar  
The approved original version of this thesis is available in print at TU Wien Bibliothek.

# Erklärung zur Verfassung der Arbeit

Alexander Gube, BSc

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Ich erkläre weiters, dass ich mich generativer KI-Tools lediglich als Hilfsmittel bedient habe und in der vorliegenden Arbeit mein gestalterischer Einfluss überwiegt. Im Anhang „Übersicht verwendeter Hilfsmittel“ habe ich alle generativen KI-Tools gelistet, die verwendet wurden, und angegeben, wo und wie sie verwendet wurden. Für Textpassagen, die ohne substantielle Änderungen übernommen wurden, haben ich jeweils die von mir formulierten Eingaben (Prompts) und die verwendete IT- Anwendung mit ihrem Produktnamen und Versionsnummer/Datum angegeben.

Wien, 13. Oktober 2025

---

Alexander Gube



Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar  
The approved original version of this thesis is available in print at TU Wien Bibliothek.

# Danksagung

Mein Dank gilt meinen Betreuern Georg Gottlob und Bernhard Krüpl-Sypien, die stets flexibel, sowohl vor Ort als auch online, verfügbar waren und immer ein konstruktives Feedback für mich parat hatten. Besonders dankbar bin ich für die zur Verfügung gestellte Flexibilität im Bereich der angewendeten Technologien. Dadurch konnte ich meinen Ideen freien Lauf lassen, diese Masterarbeit nahezu vollständig nach meiner Vorstellung umsetzen und mich persönlich in einem, wie ich finde, hochinteressanten Forschungsfeld weiterentwickeln und spezialisieren. Dieser Dank gilt auch der TU Wien für dieses interessante Masterstudium und die bereitgestellte Infrastruktur, die für meine rechenintensiven Methoden notwendig war.

Selbstverständlich möchte ich mich auch bei meiner Familie bedanken, die mir dieses Studium ermöglichte. Zusätzlicher Dank gilt meiner Schwester für ihre juristische Expertise. Für die oft kurze, aber sehr schöne Zeit, die mir zwischen den intensiven Arbeitsstunden sehr viel Kraft gegeben hat, möchte ich mich bei meiner Freundin bedanken.



Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar  
The approved original version of this thesis is available in print at TU Wien Bibliothek.

# Acknowledgements

My thanks go to my supervisors Georg Gottlob and Bernhard Krüpl-Sypien, who were highly flexible for both on-site and online meetings, in which they provided me with constructive feedback. I am particularly thankful for the offered flexibility in terms of applied technologies. This allowed me to explore my ideas freely and to create this master's thesis almost exactly as I imagined. Furthermore, it allowed me to develop and to specialize in an exciting research field. I would also like to thank TU Wien for offering this interesting master's program and providing the necessary infrastructure for my computationally intensive approaches.

Of course, I also want to thank my family, who made this study possible for me. Additional thanks go to my sister for her legal expertise. I want to thank my girlfriend for the short but wonderful time that gave me a lot of strength between the intensive work hours.



Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar  
The approved original version of this thesis is available in print at TU Wien Bibliothek.

# Kurzfassung

Durch das Lösen von Aufgaben, die menschenähnliche Intelligenz erfordern, haben Large Language Models (LLMs) viel Aufmerksamkeit erlangt. Sie haben damit auch ein enormes Potenzial, Rechtsexperten bei vieler ihrer Aufgaben zu unterstützen. Allerdings gibt es einen kritischen Aspekt, der dies bislang verhindert, nämlich Halluzination. Sie bezeichnet überzeugend formulierte Aussagen, die allerdings ohne faktische Basis sind. Hinzu kommt, dass Lösungen von LLMs oft nicht nachvollziehbar sind. Diese zwei Herausforderungen stellen die zugrundeliegenden Motive dieser Arbeit dar, die danach strebt, halluzinierte Konsequenzen zu vermeiden, die in generierten Lösungen einer rechtlichen Multi-Hop-Reasoning-Aufgabe enthalten sind. Während der Evaluierung wenden wir diese Aufgabe auf Rechtsverträge an. Mit *Chain-of-Regulation* stellen wir eine neue Methode zur Vermeidung von Halluzination vor, die Decomposed Prompting (Decomp) mit dem Verbessern von Zwischenergebnissen kombiniert. Konkret wird die Reasoning-Aufgabe in eine Reihe von Unteraufgaben modularisiert, wodurch deren Zwischenergebnisse offengelegt werden. Während diese von Decomp einfach an die folgenden Unteraufgaben weitergegeben werden, erkennen wir in der Offenlegung der Zwischenergebnisse ein hohes Potenzial, da damit die Anwendung von Self-Refinement-Methoden auf zwischenzeitliche Ergebnisse während des gesamten Lösungsvorganges ermöglicht wird. Die Evaluierung dieser Methode basiert auf einer präzisen Definition von Halluzination, um diese in unserer referenzfreien Umgebung detektieren zu können. Wir verwenden Chain-of-Thought als primäre Vergleichsbasis. Zusammengefasst liefert diese Arbeit drei zentrale Beiträge: (1) Mit *Chain-of-Regulation* stellen wir eine neue Methode vor, um Halluzinationen zu reduzieren. (2) Wir entwickeln eine Komponente, die Halluzinationen im Kontext der Multi-Hop-Reasoning-Aufgabe erkennt. (3) Wir verwenden Reasoning-Bäume, die den Reasoning-Prozess eines Modells visualisieren, um die Nachvollziehbarkeit der Antworten zu verbessern und die Entwicklung von Halluzination besser zu verstehen. Unser wichtigstes Ergebnis ist, dass im Kontext der rechtlichen Multi-Hop-Reasoning-Aufgabe *Chain-of-Regulation* fast dreimal effektiver als Chain-of-Thought ist und nur zu einem moderaten Anstieg der verwendeten Ressourcen führt.



Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar  
The approved original version of this thesis is available in print at TU Wien Bibliothek.

# Abstract

Large Language Models (LLMs) have gained immense popularity for their ability to solve tasks that require near-human intelligence. Hence, they also have enormous potential to assist experts in various tasks within the legal domain. However, there is a significant reason that currently prevents this, namely, hallucination, which refers to reasonable-sounding statements lacking a factual basis. Furthermore, hallucination is usually accompanied by a lack of explainability of LLM responses. These two challenges motivate this thesis, which aims to mitigate hallucinated consequences contained in solutions to a multi-hop legal reasoning task. During evaluation, we apply this task to a set of legal contracts. We propose a novel hallucination mitigation technique, *Chain-of-Regulation*, which combines Decomposed Prompting (DecomP) with the refinement of intermediate results. Specifically, the reasoning task is decomposed into a sequence of subtasks, thereby exposing subtask results. While they are transferred to the following subtasks by DecomP, we identify the exposure of intermediate subtask results as an excellent opportunity to improve them by applying self-refinement strategies during the whole reasoning process. To evaluate our approach, we employ a concise definition of hallucinations that enables us to detect them in a reference-free setting. We use Chain-of-Thought as the main baseline. In summary, our work makes three key contributions: (1) With *Chain-of-Regulation*, we propose a new hallucination mitigation technique. (2) We develop a hallucination verification component to identify hallucinated facts in the context of multi-hop reasoning. (3) We use reasoning trees, which are a visualization of the model's reasoning process, to enhance response explainability and to understand the development of hallucination better. The key finding is that in the context of the multi-hop legal reasoning task, *Chain-of-Regulation* is almost three times as effective as Chain-of-Thought in mitigating hallucination, with only a moderate increase in resources.



Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar  
The approved original version of this thesis is available in print at TU Wien Bibliothek.

# Contents

<b>Kurzfassung</b>	<b>xi</b>
<b>Abstract</b>	<b>xiii</b>
<b>Contents</b>	<b>xv</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Aim of the Thesis . . . . .	2
1.2 Contribution . . . . .	4
1.3 Structure of the Thesis . . . . .	4
<b>2 Preliminaries</b>	<b>5</b>
2.1 Fundamental Principles . . . . .	5
2.2 Definition of Task . . . . .	11
2.3 Formalization of Task Context . . . . .	17
<b>3 Related Work</b>	<b>23</b>
3.1 Hallucination Mitigation Strategies . . . . .	23
3.2 Hallucination Detection & Evaluation Strategies . . . . .	28
<b>4 Methodology</b>	<b>31</b>
4.1 Environmental Setup . . . . .	32
4.2 Task Decomposition . . . . .	42
4.3 Intermediate Regulation . . . . .	57
4.4 Hallucination Detection . . . . .	65
4.5 Hallucination Development & Propagation . . . . .	71
<b>5 Evaluation</b>	<b>75</b>
5.1 Evaluating the Grounder . . . . .	75
5.2 Trade-Off between Hallucination Mitigation & Relevance . . . . .	77
5.3 Evaluating Hallucination Mitigation . . . . .	78
5.4 Determining the Development of Hallucination . . . . .	79
<b>6 Results &amp; Discussion</b>	<b>81</b>
	xv

6.1	Grounding . . . . .	81
6.2	<i>Chain-of-Regulation</i> : Regulations × Decomposition Levels . . . . .	86
6.3	Development of Hallucination . . . . .	92
<b>7</b>	<b>Conclusion &amp; Future Work</b>	<b>97</b>
7.1	Conclusion . . . . .	97
7.2	Future Work . . . . .	98
	<b>Overview of Generative AI Tools Used</b>	<b>101</b>
	<b>List of Figures</b>	<b>103</b>
	<b>List of Tables</b>	<b>105</b>
	<b>List of Algorithms</b>	<b>107</b>
	<b>Bibliography</b>	<b>109</b>

# Introduction

Large Language Models (LLMs) have gained enormous popularity over the past years for their ability to process and generate natural language. Famous benchmarks such as Massive Multitask Language Understanding (MMLU) [HBB<sup>+</sup>21] document their impressive results in tasks that were recently considered to require near-human intelligence. This benchmark utilizes a set of 57 tasks from various domain-specific fields, including computer science and law, in multiple languages. It also illustrates the rapid improvement of LLMs. As of 2025, the average performance on MMLU has improved steadily over the past five years and is expected to saturate soon, such that there are ambitions to replace it in favor of more difficult benchmarks [MFP<sup>+</sup>25].

The interaction with LLMs is convenient for people as they interact with it via natural language, or in the case of multimodal models, in the form of images, videos, or speech. As a result, LLMs and other AI tools have become part of the everyday life of many people. However, even more consequential is the usage of LLMs in many professional areas, especially critical domains. Take healthcare for example, where LLMs have been adopted to support medical diagnostics, interpret imaging data in radiology, or provide clinical decision support [KKK24]. Furthermore, language models are applied in the legal domain, such as *Lawyer LLaMA* [HTZ<sup>+</sup>23], which is fine-tuned with Chinese legal knowledge to assist in legal judicial tasks.

Although these applications are convenient for their users, language models are trained to generate text rather than to generate only factual, correct text, and hence usually tend to come up with non-factual information that sounds convincing even in cases of missing knowledge. This is generally referred to as hallucination [Gum25]. In combination with the missing vigilance of users, these limitations prevent a safe usage of LLMs in critical areas such as the two domains mentioned. To address this issue, numerous approaches have been suggested. For instance, *Lawyer LLaMA* addresses this issue by the retrieval of additional context during generations. Apart from hallucinations, the answers of LLMs also lack explanations entirely or give explanations that do not faithfully reflect their

behavior [TMPB23]. This results in missing traceability, i.e., users do not understand the solution approach. However, this is also a highly desirable characteristic to validate solutions, which would also help to restore the trust undermined by hallucinations.

Besides retrieving additional context during generation, another efficient approach to mitigate hallucination is *Decomposed Prompting* [KTF<sup>+</sup>23]. The former method is suitable for closing domain-specific knowledge gaps, while decomposition is an effective strategy to reduce hallucination emerging in answers to complex tasks. Specifically, decomposition reduces an initial task into a sequence of simpler subtasks, where subsequent subtasks utilize intermediate results. Thus, the sequential execution of these subtasks results in a solution to the initial task.

By breaking the monolithic generation process of a complex task, decomposition forces the model to solve the task according to a predefined sequence of subtasks, thus exposing intermediate solution states in the form of subtask results. These intermediate solution states would otherwise be entirely transparent to the user, as they cannot be accessed or manipulated in a single, monolithic model inference. Hence, the decomposition of a task creates an opportunity to both observe and intervene in the solution process of an LLM for this task. Therefore, to further reduce hallucination, it seems reasonable to improve the subtask results by injecting suitable intermediate techniques to optimize the subtask results such that the hallucination contained in the final answer is minimized. The mitigation of hallucinations by applying this concept is the primary motivation for this work. In addition, the access to the subtask results, which we refer to as regulation interfaces, allows us to analyze and quantify hallucination so that we can observe the propagation of hallucination. We develop and evaluate this approach in a domain-specific setting, namely the legal domain.

### 1.1 Aim of the Thesis

The fundamental hypothesis, which constitutes the foundation of this thesis, claims that the standard monolithic generation process of LLMs not only favors the emergence of hallucination, specifically for complex tasks, but it additionally prevents efficient hallucination mitigation techniques in between. To overcome this limitation, we propose the *Chain-of-Regulation* approach, which aims to utilize the regulation interfaces to improve intermediate task results with the purpose of mitigating hallucination in the final task solution. Accordingly, we can derive the following fundamental research question that we further divide into four subgoals.

- 1 To what extent can we reduce hallucination contained in an LLM's solution to a complex task if we optimize the intermediate results of subtasks into which the initial task was decomposed?
  - 1.1 To what extent can we decompose the task for the purpose of intermediate hallucination mitigation?

- 1.2 How can we improve intermediate outputs such that hallucination in the solution to the initial task is reduced?
- 1.3 To what degree can we detect and quantify hallucination in the task result?
- 1.4 How does hallucination develop in a setting that involves multi-hop reasoning?

The application and evaluation of our proposed approach is restricted to a specific task in the legal domain. Specifically, given a legal contract, a concrete scenario, and a set of relevant Austrian federal laws, the task is to derive legal consequences that follow from the laws and the contract in the context of the scenario. Because of its complexity, we aim to simplify the legal domain to a highly formalizable system composed of a set of implication rules, which are if-then rules to model legal norms, and a set of basic factual statements, which describe real-world scenarios. As such, we abstract the described task to a multi-hop reasoning problem, which refers to possible dependencies between implication rules. For example, a legal consequence that follows from a specific scenario may result in another legal consequence. Due to the absence of a universally accepted definition for hallucination in the literature that we can apply, the aim of this work also includes a specification of hallucination in the context of the task formalization. Consequently, answers to the above questions are always given in the context of these definitions, which are described in Chapter 2.

The first subgoal is to explore different decompositions of the initial task in terms of their effectiveness in reducing hallucinations when applied in combination with intermediate regulation. Specifically, the decomposition transforms the monolithic prompt that describes the initial task into a series of subtask prompts that are then answered in a sequential generation process. Then, we aim to compare different prompt-based methods to reduce hallucination and apply them to the intermediate task results exposed by the decomposition. This allows us to answer to what extent intermediate regulation achieves hallucination mitigation. Hence, *Chain-of-Regulation* is effectively a set of approaches that arises from the combination of decompositions and regulation strategies that we explore.

The third milestone aims to evaluate this set of approaches in terms of hallucination reduction. Therefore, the goal is to develop an LLM-based component that detects hallucination, which is based on our definition. This is the subgoal required to allow a final evaluation of the solutions to the root problem of the thesis. Finally, as already mentioned, the task decomposition allows us to capture intermediate subtask results, which can be used to observe the development and propagation of hallucination. As a result, we ask which methods there are to describe this in our multi-hop reasoning setting. This subgoal is not strictly required to answer the overall research question. However, an answer to this subquestion helps to improve our understanding of the emergence and propagation of hallucination in our setting.

## 1.2 Contribution

The key contributions of this work to the state of the art are listed below.

1. By combining task decomposition with intermediate hallucination mitigation, this thesis proposes a novel approach, namely *Chain-of-Regulation*, to mitigate hallucination in a highly formalizable legal setting.
2. Based on the concept of *Chain-of-Regulation*, we develop and evaluate a set of approaches specialized in solving a multi-hop reasoning task to derive legal consequences that follow from a legal contract or relevant Austrian federal laws in the context of a concrete scenario.
3. We provide a definition for hallucination in a highly formalizable legal setting and develop a component specialized in detecting such hallucinations.
4. This thesis utilizes the access to intermediate solution states to analyze the development and propagation of hallucination in the task context. As a result, it contributes to understanding how hallucination emerges and propagates in this setting. Furthermore, the access to the intermediate results contributes to better response explainability.

## 1.3 Structure of the Thesis

The first section of Chapter 2 gives a brief introduction to fundamental principles that form the foundation of the development and success of Large Language Models, before we summarize important findings on hallucination from the literature. Then, we give a formal definition of the legal task that constitutes the setting of this thesis, its context, and a formal definition of our notion of hallucination.

In Chapter 3, we give an overview of existing methods to mitigate hallucination and means to detect it, which are important to evaluate approaches in terms of hallucination reduction. These methods are, of course, not limited to our notion of hallucination but rather address various problems faced with LLMs that are usually summarized under the umbrella term hallucination. In this context, we also explain what makes our approach different from these existing approaches.

Chapter 4 explains the detailed approach to answer the research question according to the defined subquestions. Apart from that, we also document technical details such as basic infrastructure setup, prompt design, and the RAG system that was built to retrieve relevant Austrian law. In Chapter 5, we explain how we evaluate our approach. In addition to the actual evaluation, this chapter also documents the strategy to examine the evaluation component itself.

Chapter 6 describes detailed results for each subgoal and discusses the results. Based on these findings, Chapter 7 concludes this work and proposes future work.

# Preliminaries

This chapter introduces concepts that we consider to be fundamental for this work in Section 2.1. In essence, we introduce large language models and one of their main issues, namely, hallucination. Apart from this, we specify the multi-hop legal reasoning task in Section 2.2, which defines the setting in which we develop and evaluate *Chain-of-Regulation*. Finally, based on this definition, Section 2.3 introduces the formalization of this setting, which allows us to define hallucination in this context.

## 2.1 Fundamental Principles

A basic knowledge of large language models and hallucination is fundamental for this thesis. Therefore, we give a brief introduction to the required preliminary knowledge below. We explain the theoretical foundations of LLMs before we address their architecture and the underlying training phases. We then introduce hallucination, the mitigation of which is the primary motivation of this work. We summarize findings from the literature, including known forms of hallucination and important causes of it. Finally, we underline why it is a problem, especially in the legal domain.

### 2.1.1 Large Language Models

Artificial Intelligence (AI) is commonly separated into symbolic and sub-symbolic methods [IK20], where LLMs are part of the latter. The former approach employs symbols, such as those of first-order logic, to explicitly represent knowledge, which is then typically applied using explicit logical rules in a deductive manner. On the contrary, sub-symbolic methods capture the correlation between input data and output data using mathematical or statistical methods. As such, they usually require more data and are less interpretable than symbolic methods. LLMs are based on the following fundamental idea: *How well can the next letter of a text be predicted when the preceding  $N$  letters are known* [Sha51].

In particular, they model language using a probability distribution of (sub-)words, which is why they are considered to be sub-symbolic.

First, we introduce important mathematical preliminaries on which LLMs are based [XZ25]. The probability of a sequence of words  $(x_1, x_2, \dots, x_n)$  can be represented by conditional probabilities as shown in Equation 2.1. Hence, the probability of a sequence of words is the product of the conditional probabilities of each word given its preceding words.

$$P(x_0, \dots, x_n) = \prod_{i=0}^n P(x_i | x_0, \dots, x_{i-1}) \tag{2.1}$$

The underlying idea of LLMs is to model this probability using deep neural networks, which are known to be robust function approximators [HLP<sup>+</sup>25]. Then, a trained neural network parameterized by  $\theta$  can be used as shown in Equation 2.2 to predict the next word  $\hat{x}$  of a sequence. It decodes the word with the highest probability according to the model in vocabulary  $L$ , which is the set of available words. Decoding words one by one based on their predecessors is referred to as autoregressive word generation.

$$\hat{x} = \arg \max_{x_i \in L} P_\theta(x_i | x_0, \dots, x_{i-1}) \tag{2.2}$$

This idea is utilized by what is denoted as decoder-only architecture. On the contrary, encoder-decoder models transform input text into a hidden internal representation, the hidden space [FLY<sup>+</sup>23]. Then, the decoder generates the output words based on the preceding output words and the internal representation, while the decoder-only architecture simply takes all preceding words into account. Figure 2.1 illustrates this difference. As the underlying model of this work, Gemma 3, follows a decoder-only architecture [Tea25], this introduction focuses on them.

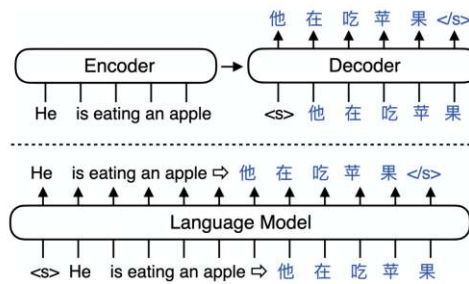


Figure 2.1: An illustration of how tokens are processed in an encoder-decoder architecture and a decoder-only model [FLY<sup>+</sup>23].

Based on these foundations, we can now have a more detailed look at the internal mechanisms employed by decoder-only architectures. First, it is important to note

that LLMs do not process probabilities of words as previously presented, but rather operate on tokens. Therefore, the first important step is to transform the raw input text into a sequence of tokens, e.g., by splitting the text into single words, which is referred to as tokenization. While tokenizing on a word basis is intuitive, in practice, subword-based tokenizations such as SentencePiece [KR18], which is used by Gemma, are preferred, due to better performance and more efficiency [WSV<sup>+</sup>24]. Then, tokens, which are represented by integers, are transformed into high-dimensional dense vector representations, which are referred to as embeddings. Besides mapping the sequence to a shape that neural networks can efficiently process, embedding models preserve the semantic meanings of tokens to some extent [vdBP24].

Both tokenization and embedding can be considered as pre-processing steps before the actual task of the decoder-only LLM, namely, the next token prediction, is carried out by its deep neural networks. To achieve this, they incorporate a crucial mechanism, namely self-attention, which is a key concept of the Transformer architecture [VSP<sup>+</sup>23]. Initially, the attention mechanism was introduced to overcome the limitation of conventional sequence-to-sequence models in processing long sequences [BCB16]. Its core idea is to model relationships between tokens within a sequence and to weight them accordingly, which is more effective than weighting all of them equally.

The training of an LLM is usually split into two phases, the pre-training and the fine-tuning [XZ25]. During the pre-training, also referred to as foundation training, the goal is to adjust the model parameters  $\theta$  such that it models the underlying causal token probability distribution  $P_\theta$  of a target language as accurately as possible. As is common with neural networks, the model is trained to approximate this distribution by minimizing a loss function in a training dataset  $T$ . This optimization problem is usually approached using gradient descent. In this case, the objective is to maximize the log-likelihood of the training sequences  $\mathbf{x}$ , given as  $L_\theta(\mathbf{x})$  in Equation 2.3.

This optimization process is modeled by Equation 2.4, which asks for the best parameterization  $\hat{\theta}$  of the model with reference to  $T$ . The pre-training of an LLM requires a lot of computational resources as it tunes billions of parameters, and the number of used tokens is on the order of  $10^{12}$  in the case of Gemma 3 [Tea25].

$$L_\theta(\mathbf{x}) = \sum_{i=1}^n \log P_\theta(x_i | x_0, \dots, x_{i-1}) \quad (2.3)$$

$$\hat{\theta} = \arg \max_{\theta} \sum_{\mathbf{x} \in T} L_\theta(\mathbf{x}) \quad (2.4)$$

Once the pre-training of an LLM is completed, it is common to optimize its outputs with regard to a specific task or to extend its capability with domain-specific knowledge, which is referred to as fine-tuning [PZKS24]. For instance, to create an instruction-tuned model, a typical approach is to use another dataset with pairs of desired inputs and

outputs, which is usually a lot smaller than the dataset used during pre-training. Again, the model parameters are adjusted to optimize the loss for this data.

In addition, reinforcement learning from human feedback (RLHF) is an effective method to train an LLM with human objectives, which is frequently referred to as alignment [KWBH24]. Humans manually evaluate the outputs returned from a model in terms of specific objectives. This feedback is then used by reinforcement learning algorithms to further optimize the model's parameters.

### 2.1.2 Hallucination

Even though they can be compelling, it is crucial to remember that LLMs are designed to predict probable sequences. Fine-tuning, specifically alignment, is intended to teach models human values, including trustworthiness. However, they still tend to produce plausible but inconsistent or false answers, which is referred to as hallucination [MMN<sup>+</sup>25]. This is a significant limitation of LLMs, which also prevents their safe application in critical domains.

There are ambitions to define and characterize hallucination in this highly active field, allowing us to summarize popular approaches. First, it is important to note that there is no universally accepted definition of hallucination [JLF<sup>+</sup>23], but rather several different notions depending on the perspective. For instance, in formal settings, hallucination is frequently defined as deviation from a ground-truth function [XJK25]. In contrast, conceptually, hallucination is usually described as the generation of unfaithful, nonsensical, or fabricated statements [Cos25, HYM<sup>+</sup>25]. In addition, in case of such outputs, LLMs usually do not indicate that their answers are fabricated. This is critical, as it leads to a lack of confidence in LLM answers.

In terms of taxonomy, several works [Cos25, HYM<sup>+</sup>25, Gum25] suggest a division into the two following types of hallucination:

- **Intrinsic Hallucination.** This type of hallucination refers to inconsistencies between the model's input and its output. For example, take a summarization task with an input containing "Austria is part of the EU and its capital is Vienna." for which an LLM generates a summary saying "Brussels is the capital of Austria.". In this case, the output directly contradicts the input. This type of hallucination indicates the model's inability to guarantee consistency during the generation.
- **Extrinsic Hallucination.** If the generated output does not contradict the input directly and is also not verifiable by any available knowledge source, it is considered to be extrinsically hallucinated. Intrinsic hallucination can be easily detected, while it is difficult or may even be infeasible to verify if an output is either factual or contains extrinsic hallucination. This phenomenon is caused by missing knowledge required to solve a task, which can result from inadequate training data or the model's limitation to capture this knowledge.

Further classifications include more pragmatic approaches that divide hallucination into, for example, factual errors, logical inconsistencies, temporal disorientations, ethical violations, etc. [Cos25].

When it comes to the causes of hallucination, there exist numerous categorizations [Cos25, HYM<sup>+</sup>25, JLF<sup>+</sup>23], which we combine to provide an overview summarized in Figure 2.2. It is important to note that we use this overview only for better visualization, as most of the identified causes are interrelated.

First, in terms of **model and inference** we can identify the auto-regressive behavior of LLMs as a source of hallucination, as models are designed to predict probable tokens instead of focusing on factual accuracy. In addition, the majority of decoding strategies rely on randomness to escape what is known as the *likelihood trap*, i.e., in practice, tokens that do not have the highest probability are also generated [HYM<sup>+</sup>25]. This randomness increases the tendency to hallucinate. Furthermore, models can suffer from overconfidence, such that they respond with high confidence even if they are unfaithful. This is caused by the self-attention mechanism, which tends to prioritize recent words, thereby neglecting previous context, and hence focuses on text fluency rather than consistency. This is again related to its auto-regressive nature. An insufficient knowledge representation stems from the model’s incapability to represent the knowledge required to solve its tasks [Cos25], e.g., because of a poor model architecture or inappropriate training data.

In terms of **training**, hallucination can be induced by misalignment, which refers to undesired model behavior acquired during alignment [Cos25]. For instance, the model prioritizes maximizing the reward it gains from human feedback over maintaining truthfulness. During training, an LLM represents knowledge in its parameters, which is referred to as parametric knowledge. When solving tasks, models tend to neglect the provided context in favor of this knowledge, which can contribute to hallucination [LPC<sup>+</sup>21]. The discrepancy of the decoding context between training, where this context is part of the ground-truth data samples, and the actual inference, where it is based on previously generated tokens, is another potential source of hallucination, especially for long outputs due to cascading errors. This is referred to as exposure bias [JLF<sup>+</sup>23].

The **data** used during training can also be a source of hallucination. Despite their enormous size, training datasets have a knowledge boundary, which restricts the provided knowledge, especially for specific domains [Cos25]. In addition, LLMs are trained once on a static dataset so that their knowledge has a cutoff date. Furthermore, the quality of the training data is crucial in minimizing hallucinations. Specifically, datasets can suffer from misinformation such as fake news and several types of biases, especially social bias stemming from social media platforms [HYM<sup>+</sup>25]. Without data sanitation, they are acquired by LLMs during their training.

This analysis raises the question of whether hallucination can be eliminated, provided that the identified issues are effectively resolved. In fact, using a formal setting, where hallucination is defined as a mismatch of the generated outputs with a ground-truth function, it can be shown that hallucinations are inevitable if LLMs are used as general

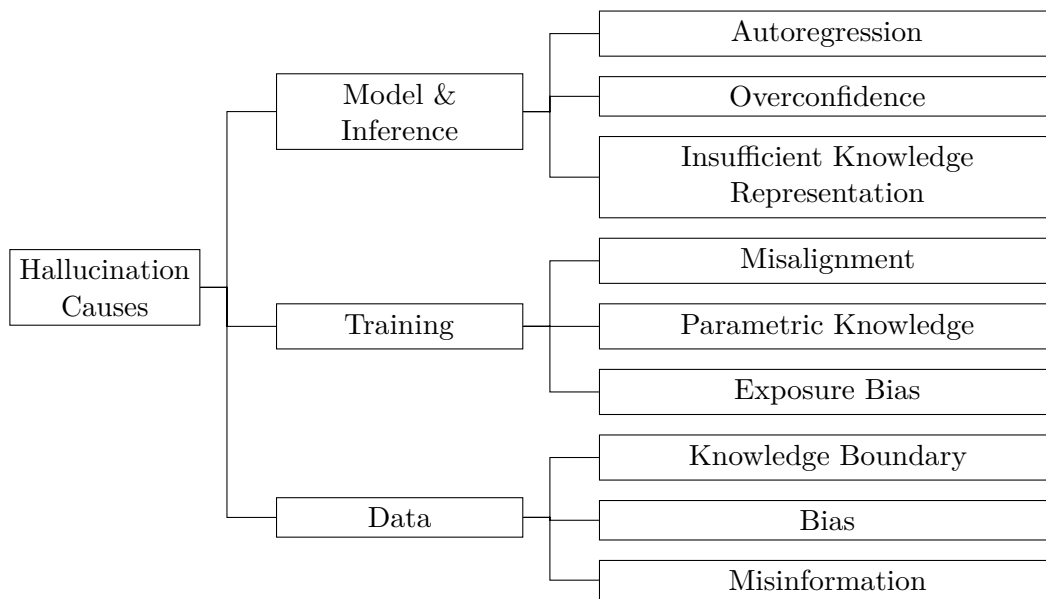


Figure 2.2: A taxonomy describing important causes of hallucination [Cos25, HYM<sup>+</sup>25, JLF<sup>+</sup>23].

problem solvers [XJK25]. In other words, they conclude that hallucination cannot be eliminated by solely improving the used training data, learning algorithms, or model design, as it is rather an innate property of any large language model. From this observation, we can further conclude that hallucination is a sustainable challenge and strategies to mitigate it, including *Chain-of-Regulation*, will always be necessary for the safest possible use of LLMs.

As shown in [DMSH24], the analysis of hallucination is highly relevant for the legal domain. They profile hallucination in this critical domain based on a set of legal tasks, which are all based on a direct, verifiable question about a federal court case in the US. Hallucination is evaluated using a fuzzy matching between the expected answer and the given output for tasks with ground-truth, whereas self-consistency is used in the absence of reference solutions. Self-consistency as a strategy to detect hallucination is introduced in Section 3.2. This work reports alarmingly high amounts of hallucination in the legal domain. Figure 2.3 gives the mean hallucination rate for different LLMs over the case prominence. As expected, the more prominent the cases, the less hallucination is generated. Hence, the knowledge boundaries of these models seem to be a major contribution to hallucination. In general, these high levels of hallucination emphasize that it indeed is a major concern in the legal domain.

The detection and mitigation of hallucinations are the motivating challenges of this thesis. Section 3.2 gives an overview of important hallucination detection strategies, while in Section 3.1 we summarize existing methodologies to overcome this fundamental limitation of LLMs. In terms of this work, we focus on the mitigation of extrinsic hallucination,

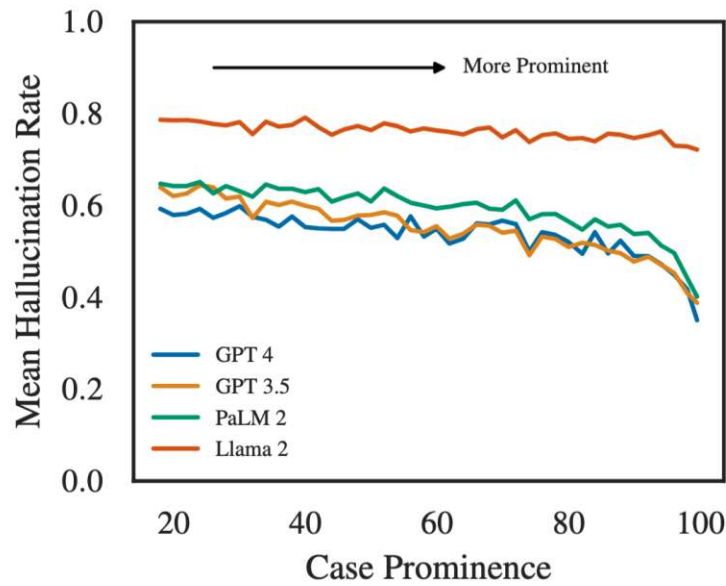


Figure 2.3: The hallucination rate of LLMs over the prominence of cases addressed by the tasks. The prominence of a case is measured by PageRank percentile. The decline of hallucination rates with increasing prominence indicates a knowledge boundary [DMSH24].

which we further restrict by using a definition of hallucination given in Section 2.3. Essentially, it is based on the notion of grounded facts, i.e., statements that are supported by reference texts.

## 2.2 Definition of Task

This section contains the detailed definition of the legal task that constitutes the context for which the approach of this work is developed and evaluated. First, we describe the setting of the task, which is composed of a contract, relevant laws, and a concrete scenario on which both are applied. Based on this, we then define the actual task. Furthermore, as the task is part of the complex legal domain, we define important constraints that are required to focus on the core issue of this thesis, namely, the hallucination reduction using *Chain-of-Regulation*. Finally, we justify its relevance and connection to the real world.

### 2.2.1 Setting

The motivation for this task is a set of non-disclosable legal contracts from two specific areas: so-called coexistence agreements and employment contracts, primarily between a company and its CEO. In the context of trademarks, coexistence contracts are a common

tool in case there are at least two companies that own similar marks with the potential of confusion [UK 08]. To avoid infringing on each other's rights, they agree on the conditions under which they intend to use their marks. Likewise, this type of contract is common for intellectual property in general. The main purpose of these contracts is to avoid legal consequences.

Apart from individual specifications, the available coexistence agreements contain the following typical sections:

1. A precise description of the parties to the contract, typically two companies
2. A precise description of the trade marks or other intellectual property for which this agreement is made
3. Restrictions and obligations on the usage of the rights that are associated with the intellectual property
4. The place of jurisdiction for the contract

Employment contracts, in the Austrian legal context denoted as *Angestelltenvertrag*, regulate the rights and obligations between an employer and a salaried employee. As such, they are typically entered into by a company and its employees. Specifically, our dataset contains chief executive officer employment agreements (*Geschäftsführervertrag*), which, in addition to the terms of a conventional employment contract, typically include special provisions related to the CEO's exclusive role, such as a non-compete clause. Therefore, apart from individual specifications, the available CEO contracts contain the following typical sections:

1. A precise description of the parties to the contract, typically a company, the employer, and a natural person, the employee
2. Terms that regulate general employment aspects, including salary, scope of duties, vacation, sick leave, company car, and place of work
3. A confidentiality clause (*Geheimhaltung*) that obliges the CEO not to expose information about business or operational processes of the company that becomes known to the CEO during job activities, e.g., financial details.
4. A non-compete obligation (*Wettbewerbsverbot*, *Konkurrenzklause*) that disallows the employee to work in the same business area as the company during the employment and a specified period thereafter
5. A clause regarding the intellectual property laws (*Immaterialgüterrechte*), which typically specifies that the intellectual property of the CEO that emerged during the employment is transferred to the company

## 6. The place of jurisdiction for the contract

As the original versions of the contracts contain information that is sensitive to both companies and humans, they are provided to us in anonymized form. As such, names of companies and trade marks are usually replaced by pseudonyms (e.g., 'A', 'Lz'), whereas names of humans are either also replaced by pseudonyms or removed entirely. When it comes to the applicability of the contracts to the scenarios, we assume pseudonyms to be uncritical as long as they are also consistently used in the scenarios. However, for entirely removed names, we manipulate the contracts by injecting generated names.

In addition to the contracts, the next resource required for the task is the relevant laws that can be applied in the context of the contract. The majority of the available contracts are formulated in the Austrian legal context. Therefore, we restrict the legal application to Austrian federal laws, which can be retrieved from the official government source, the *Rechtsinformationssystem des Bundes* (RIS) <sup>1</sup>. Specifically, we use consolidated federal law, which means that all amendments and corrections are incorporated into the original version of a legal document. The unit of legal text that the RIS provides is a paragraph, an article, or an annex to a legal norm. To create the context of the task, we feed a configurable number of such legal text units to the model. The following text quotes § 1 *Markenschutzgesetz 1970* <sup>2</sup> and demonstrates how legal texts are used in the context of the task.

Kurztitel  
 Markenschutzgesetz 1970  
 Kundmachungsorgan  
 BGBl. Nr. 260/1970 zuletzt geändert durch BGBl. I Nr. 91/2018  
 Typ  
 BG  
 §/Artikel/Anlage  
 § 1  
 Inkrafttretensdatum  
 14.01.2019  
 Index  
 26/02 Marken- und Musterschutz

Text  
 I. ABSCHNITT  
 Allgemeine Bestimmungen  
 § 1. Marken können Zeichen aller Art sein, insbesondere Wörter, einschließlich  
 Personennamen, oder Abbildungen, Buchstaben, Zahlen, Farben, die Form

<sup>1</sup><https://www.ris.bka.gv.at>

<sup>2</sup><https://www.ris.bka.gv.at/Dokumente/Bundesnormen/NOR40209561/NOR40209561.html>

oder Verpackung der Ware oder Klänge, soweit solche Zeichen geeignet sind,  
1) Waren oder Dienstleistungen eines Unternehmens von denjenigen anderer Unternehmen zu unterscheiden und  
2) im Markenregister in einer Weise dargestellt zu werden, dass die zuständigen Behörden und das Publikum den Gegenstand des ihrem Inhaber gewährten Schutzes klar und eindeutig bestimmen können.

### Anmerkung

Siehe dazu auch Art. 6 quinquies lit. C Abs. 1 Pariser Verbandsübereinkunft zum Schutz des gewerblichen Eigentums, BGBl. Nr. 385/1969.

### Schlagworte

Wortmarke, Bildmarke, Wort-Bild-Marke, Fabriksmarke, Handelsmarke

Zuletzt aktualisiert am

27.12.2018

Gesetzesnummer

10002180

Dokumentnummer NOR40209561

The typical structure of a legal norm follows the logical form of an if-then rule. Specifically, if the set of all required premises (*Tatbestand*) is fulfilled, then the legal consequences (*Rechtsfolgen*) follow. Based on this structure of legal norms, we define an important abstraction in Section 2.3.

A concrete scenario, the so-called *Sachverhalt*, in which both the contract and the relevant laws are applied, completes the setting of the task. A scenario describes a specific situation in the world that is associated with the contract, including concrete actors. Hence, in the case where the contract uses pseudonyms, these names are also used in the scenario. Furthermore, the scenario is required to allow the application of relevant laws and clauses of the contract.

In the context of this thesis, we compose scenarios based on the contracts using an LLM. However, in general, the approach developed for the legal task can also be applied to real-world scenarios, which would be the common use case. The following text summarizes one of the fictional scenarios that we use.

Die Alpenkräuter GmbH, ein Tiroler Familienunternehmen, vertreibt seit den 1990er-Jahren Teemischungen unter der Marke „Bergkräuter“, die seit 2001 als Wortmarke im österreichischen Markenregister eingetragen ist. Die Produkte werden österreichweit in Supermärkten verkauft und genießen einen hohen Bekanntheitsgrad.

Im Jahr 2025 gründet die BioNatur KG in Vorarlberg ein Start-up, das ebenfalls Kräuterprodukte vertreibt. Sie bringt unter dem Namen „BergKräuter

Natur“ Tees und Nahrungsergänzungsmittel auf den Markt. Die Verpackungen sind grün gestaltet und enthalten stilisierte Alpenblumen, wodurch sie den Produkten der Alpenkräuter GmbH optisch stark ähneln.

Kurze Zeit später beschwert sich die Alpenkräuter GmbH über sinkende Verkaufszahlen und erhält von Kunden Rückmeldungen, wonach die Produkte verwechselt würden. Das Unternehmen erhebt daher Klage gegen die BioNatur KG wegen Markenverletzung und macht zusätzlich Unterlassungs- und Schadenersatzansprüche geltend.

Details on the scenario composition can be found in Section 4.1.

### 2.2.2 Definition

Based on the description of its setting, we can now give a thorough description of the task. Based on a set of relevant Austrian federal legal norms and a contract, either a coexistence agreement or a CEO employment contract, the goal is to derive concrete legal consequences in the context of a concrete scenario. The consequence of a legal norm can be derived if it is applicable, which means that the premises of the norm are met in the context of the scenario. A consequence is concrete in our sense if it refers to the given situation.

This defines the basic legal reasoning task, which is fundamental for this work. An important aspect, which adds complexity to the task, is the multi-hop characteristic of the reasoning. That is, a legal consequence can meet the premises of another legal norm, which results in another legal consequence. This is an essential element of the task that is addressed during the task decomposition and the development of hallucination.

The retrieval of the relevant Austrian federal laws, which are used by the reasoning task to derive legal consequences, is not part of the actual task. Hence, neither decomposition nor intermediate regulation is applied in this case. As a result, all developed approaches use the same set of laws for comparison purposes. Figure 2.4 illustrates a high-level abstraction of the reasoning task, which interacts with its setting, namely the scenario, the retrieved laws, and the contract to derive legal consequences.

### 2.2.3 Constraints

Based on this task definition, we identify a set of important constraints that allow us to focus on the core issue of this thesis, namely, reducing hallucinations. First, we define contract-related constraints. We assume that the contract is valid and applicable in the associated scenario. Furthermore, the set of contracts is mixed in terms of language, such that we restrict our dataset to German contracts, which has two significant reasons. On the one hand, the majority of the contracts are formulated in German, and on the other hand, the relevant laws are only available in German.

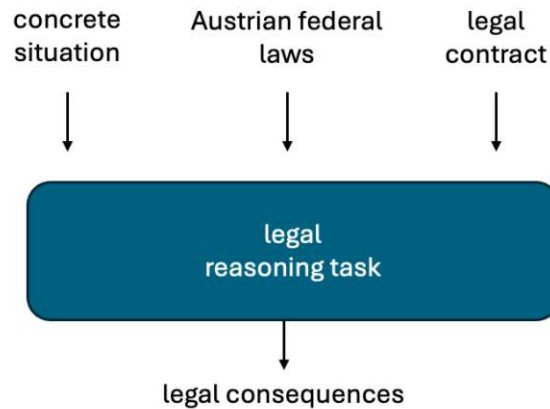


Figure 2.4: A high-level illustration of the legal reasoning task.

Apart from this, we specify a set of restrictions when it comes to the relevant legal texts. In contrast to authentic federal law, consolidated law is not legally binding. However, by using consolidated law, we avoid the challenge of determining the correct version of a law. This reduces the law retrieval complexity, which is not considered part of the actual task. In addition, we assume the retrieved legal units, e.g., paragraphs, to be self-contained, such that they do not contain references to other legal texts.

The application of Austrian law is, of course, not restricted to federal laws, but includes the Austrian federal constitution, EU law, federal laws, regulations, and decisions, in descending order of significance [Ver25]. Furthermore, there is a set of methods that ensure a consistent process during the application of the law, such as *lex specialis derogat legi generali* [Gro20], which is a common approach to resolving collisions between legal norms by prioritizing specific law over general law. As the correct application of law is not within the scope of this thesis, we restrict the task to Austrian federal law. In addition, we also neglect several legal methods. Instead, we abstract to a greedy legal policy, where legal norms are applied in a first-come, first-served manner.

#### 2.2.4 Real-World Significance

The mentioned constraints form a significant deviation from the application of law in the real world. Thus, a solution to the defined task is not a proper substitute for a lawyer. Still, it can be used as an assistive tool, particularly for contract review. Specifically, given a contract and a scenario, a solution to the task outputs legal consequences. What is more, a solution can also generate potential legal consequences if you feed it a probable or future situation. As such, it can be used to assist lawyers in reviewing contracts or to entirely automate risk analysis, which is usually the most important part of contract reviews.

## 2.3 Formalization of Task Context

The definition of the task and the restrictions associated with it, both addressed in Section 2.2, allow us to formalize the setting of the task and its expected result, the legal consequences. Based on this formalization, we can then define hallucination for this specific setting.

### 2.3.1 Setting Formalization

We abstract the complex legal domain to a highly formalized system founded on the following two concepts.

1. **Fact.** A fact that contains a concrete or general piece of information
2. **Implication Rule.** An implication rule that is composed of a set of premises and a consequence. It follows logically the structure of an if-then rule such that the consequence is derived when all premises are satisfied.

We define a fact as a statement that contains a piece of information. More specifically, we distinguish between concrete facts, i.e., those that describe a situation with entities (e.g., people) from the real world, and abstract facts, which are formulated without such entities to describe generic statements. This distinction may seem artificial at first, but it is important to distinguish concrete facts describing the scenario from premises or consequences of an implication rule, which are usually expressed abstractly. Based on this distinction, we introduce an operation that transforms abstract facts into concrete facts, which we denote as concretization. Concretizations are applied in the context of a concrete scenario. Figure 2.5 illustrates this. During concretization, generic terms (e.g., "somebody") are replaced by concrete entities (e.g., people) from the scenario.

Natural Language Inference usually defines three important relations between two facts referred to as premise and hypothesis, namely *entailment*, *neutrality*, and *contradiction* [CRLB18]. On this basis, we expand these relations to a text as premise and a fact as hypothesis. In addition, we define *containment* to describe that a fact is explicitly contained in the text without requiring additional reasoning. This distinction between *entailment* and *containment* is necessary in our setting, as implication rules model entailments that require reasoning. If a fact is contained in a text, we define it to be *grounded* by the text. In this context, we also define the *atomization* of a text, which returns the set of facts it contains. However, we neglect the definition of fact atomicity to guarantee flexibility for a model when atomizing a text.

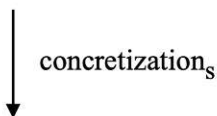
In addition, we define implication rules to model if-then rules, in our case, legal norms. Formally, they consist of a set of premises and a conclusion, where each premise and the consequence are abstract statements, i.e., facts. If for an implication rule, all concretized premises are contained in a given scenario, then the concretized consequence can be derived. As the task context is given in natural language, implication rules are also

### Scenario $s$

Fabian is employed as a baker at 'Bread&More' GesmbH and, according to his boss, always performs his work as agreed. However, at the end of the month, he doesn't receive the agreed wages.

### Abstract fact

The employer pays less than the agreed wage to the employee.



### Concrete fact

The 'Bread&More' GesmbH pays less than the agreed wage to Fabian.

Figure 2.5: An example of a fact concretization in the context of a scenario  $s$ .

provided in natural language, usually expressed as if-then rules. As such, on the one hand, they can be applied *implicitly*, which means that consequences are derived based on their representation in natural language. On the other hand, they can also be applied *explicitly*, which refers to their application based on a structured representation that utilizes the concepts of premises and consequences. Depending on the decomposition levels, as defined in Section 4.2, we employ both options. The explicit application requires a previous implication rule extraction, which outputs them in the specified structure. Figure 2.6 illustrates both application types of an implication rule.

According to this formalization, any text can be processed into a set of facts and a set of implication rules that it describes. Hence, the three components that compose the task setting from an application perspective, namely, legal norms, the contract, and the scenario, can be processed likewise. The scenario can thus be split into a set of concretized facts that describe it. A contract usually contains both facts and implication rules, while the legal texts typically only contain implication rules in the form of legal norms.

While this formalization primarily serves the definition of hallucinations in the context of this thesis, it is also utilized by the task decomposition and the grounding component. This simple reasoning model for the legal domain may also be easily generalized to other highly formalizable systems that are composed of a set of basic facts and a set of implication rules, such as physics. Although these considerations are outside the scope of this thesis, they provide an outline for future work.

## Explicit Implication Rule Application

### Scenario s

Fabian is employed as a baker at 'Bread&More' GesmbH and, according to his boss, always performs his work as agreed. However, at the end of the month, he doesn't receive the agreed wages.

### Implication Rule

$p_1$ : The employer pays less than the agreed wage.  
 $p_2$ : The employee performs his/her work properly.



c: The employee can sue or demand the difference from the employer.

### Application of Implication Rule

$p_1$ : 'Bread&More' GesmbH pays less than the agreed wage.  
 $p_2$ : Fabian performs his work properly.



c: Fabian can sue or demand the difference from the 'Bread&More' GesmbH .

## Implicit Implication Rule Application

### Scenario s

Fabian is employed as a baker at 'Bread&More' GesmbH and, according to his boss, always performs his work as agreed. However, at the end of the month, he doesn't receive the agreed wages.

### Implication Rule

If the employer performs his/her work properly and the employer pays less than the agreed wage, then the employee can sue or demand the difference from the employer.

### Application of Implication Rule

Fabian can sue or demand the difference from the 'Bread&More' GesmbH .

Figure 2.6: The application of an implication rule can be performed explicitly using its structured representation or implicitly if the rule is provided in natural language.

### 2.3.2 Definition of Hallucination

As there is no universally accepted definition for hallucination in the literature [JLF<sup>+</sup>23], we must define it for our setting to detect and mitigate hallucination, which is both required to give a proper answer to the initial research question. To achieve this, we can utilize the previous formalization of the task setting. Hence, this definition of hallucination is restricted to this specific formalization, which implies that it is agnostic to the specific domain, in this case, the legal domain, and can be reused by every setting or domain that can be abstracted to our formalization.

In the context of our formalization, we define hallucination to be the set of ungrounded facts. A fact is considered to be grounded if any of the following holds:

- The fact is grounded in the scenario. In our case, the legal contract is a part of the scenario.
- The fact is assumed. This implies that the fact is explicitly marked as an assumption in the solution's output.
- The fact is the consequence of the correct application of an implication rule.

This definition relies on an important property of our task formalization, namely that a fact can be grounded by a text (e.g., the scenario). We also allow assumptions provided that they are explicitly marked as such in the output of the solution process. Alternatively, a fact is considered to be valid if it is the consequence of an implication rule that was correctly applied. We define the application of an implication rule to be correct if all concretized premises of the rule, that are necessary to derive the consequence, are grounded. Since the formalized setting is defined as multi-hop reasoning, this definition has a recursive character. In other words, a fact that is the consequence of an implication rule can only be correctly grounded if all the respective concretized premises of this rule are grounded.

Based on this definition, we can now define an idealized grounding component that determines if a fact is grounded in the context of the task. Algorithm 2.1 illustrates this idealized program, which is the fundamental motivation of the recursive grounder that is discussed in 4.4.

**Algorithm 2.1:** Idealized Recursive Grounding *idealGround*


---

**Input:** A scenario  $s$ , a text containing all assumptions  $a$ , a set of implication rules  $IR$ , a fact  $f$  to ground

**Output:** *true* or *false*

```

1 if contains( $f, s$ ) then
2   | return true
3 end
4 if contains( $f, a$ ) then
5   | return true
6 end
7  $ir \leftarrow \text{get\_ir}(f, IR)$ 
8 if  $ir = \text{nil}$  then
9   | return false
10 end
11 for  $p \in ir.\text{premises}$  do
12   |  $p_c \leftarrow \text{concretize}(p, ir, f, s)$ 
13   | if  $\neg \text{idealGround}(p_c)$  then
14     | return false
15   | end
16 end
17 return true

```

---

The idealized algorithm uses the following idealized subroutines.

- $\text{contains}(f, t)$ . This routine takes a fact  $f$  and a text  $t$  and returns true if and only if the fact is contained in the text.
- $\text{get\_ir}(f, IR)$ . This routine takes a fact  $f$  and a set of implication rules  $IR$ , and retrieves the first implication rule whose application results in  $f$ . If there is no applicable rule, then this routine returns *nil*.
- $\text{concretize}(p, ir, f, s)$ . This routine takes a premise  $p$ , an implication rule  $ir$ , a fact  $f$ , a scenario  $s$ , and returns the concretized version of  $p$  under the assumption that  $f$  is the concretized consequence of  $ir$  in the context of scenario  $s$ .

With this definition, we can evaluate the proposed *Chain-of-Regulation* approach in terms of hallucination mitigation. Specifically, the ratio of facts that are considered to be hallucinated according to this definition is the most critical hallucination metric.



Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar  
The approved original version of this thesis is available in print at TU Wien Bibliothek.

# Related Work

In this chapter, we give an overview of previous works relevant to this thesis. We discuss which of their concepts are reused, we explain the demarcation of existing literature in relation to the *Chain-of-Regulation* approach, and thus identify the research gap that this work aims to address. This thesis explores a methodology to reduce hallucination, and therefore, we discuss existing strategies for hallucination mitigation in Section 3.1. Additionally, we propose methods to evaluate hallucination within the context of our formalization. Hence, we address existing approaches in Section 3.2.

## 3.1 Hallucination Mitigation Strategies

When it comes to the prominent challenge of hallucination mitigation, there are numerous approaches covering a huge variety of different strategies. As a result, we first give a broad overview that introduces the underlying paradigms, before we discuss methods that are similar to our approach in more detail. Hallucination mitigation strategies can be classified into Developing Models and Prompt Engineering [TZJ<sup>+</sup>24], which is illustrated in Figure 3.1.

### 3.1.1 Spectrum of Hallucination Mitigation Paradigms

In terms of *Developing Models*, we introduce fine-tuning as its most prominent member, where the parameters of a pre-trained model are adjusted by further training on a smaller dataset specialized for a specific task or domain [PZKS24]. The distinction into pre-training and fine-tuning allows for decoupling the costly pre-training, which is required for the model to develop a general language understanding, from the more lightweight fine-tuning. Hence, the same base model can be fine-tuned to a variety of different tasks.

In general, fine-tuning is applied to improve the performance for specific tasks (e.g., summarization) or to specialize the model in specific domains (e.g., legal domain), which

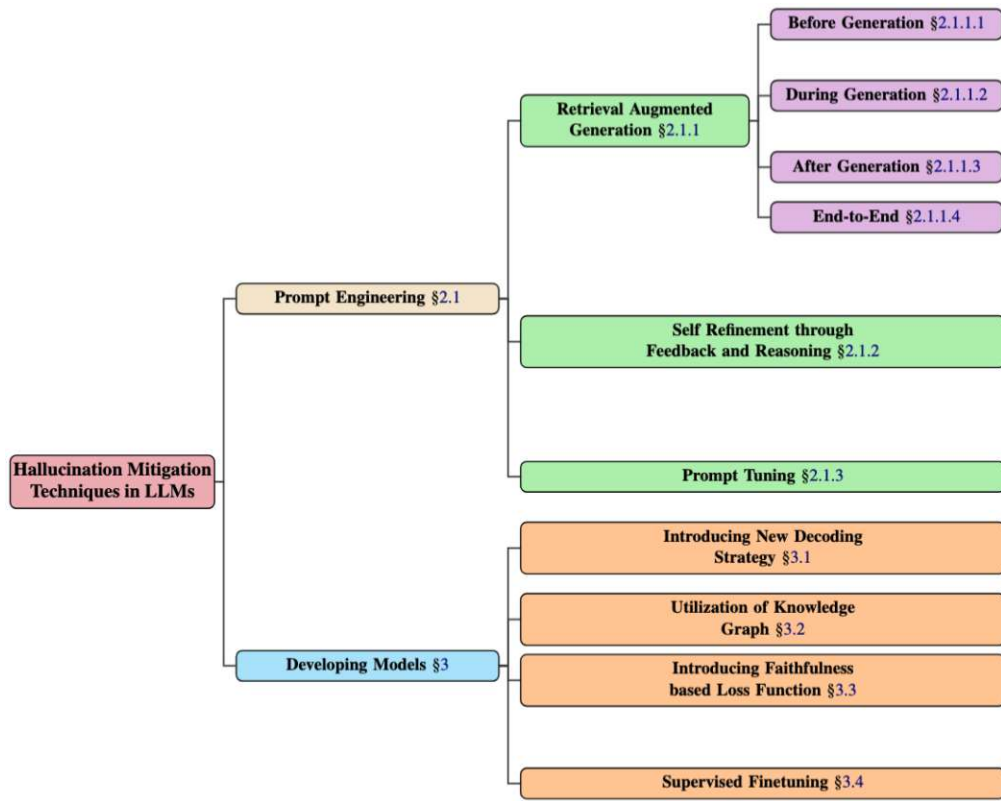


Figure 3.1: Taxonomy of hallucination mitigation strategies according to [TZJ<sup>+</sup>24]. *Chain-of-Regulation* can be seen as Self-Refinement through Feedback and Reasoning method.

are typically underrepresented in the training data of the base model [PZKS24]. As a result, an alternative approach to reduce hallucination in the legal domain would be to fine-tune a model on a respective Austrian legal text corpus. *Chain-of-Regulation* does not apply fine-tuning and instead uses ready-to-use instruction-based models. According to [MBK<sup>+</sup>22], fine-tuning in the legal domain is a necessity to overcome the difference between general language captured by the base model and the precise legal language. Thus, fine-tuning the model used in our approach is a promising strategy to improve results further. As such, this optimization is left for future work.

In comparison to the costs of pre-training, fine-tuning is cheap. However, during regular fine-tuning (parameter-efficient methods such as LoRA [HSW<sup>+</sup>21] are an exception), it is still required to adjust all parameters of the model, which results in computational costs. In addition, fine-tuning also requires access to the model’s parameters. Another hallucination mitigation paradigm that does not need either is to provide external knowledge fed to the model during inference as context. Retrieval Augmented Generation (RAG) [LPP<sup>+</sup>21] adds external knowledge, such as domain-specific corpora, to the model’s context in two stages. First, external textual knowledge is split into so-called documents

whose dense vector embeddings are computed for storage in a vector index. Then, during retrieval, a query is embedded to determine the most relevant documents using vector similarity measures, such as Maximum Inner Product Search (MIPS). These documents are then added to the LLM’s context. In summary, overcoming the knowledge gap of domain-specific areas with RAG is a popular method to mitigate hallucinations, including the legal domain [HCF25, KSR<sup>+</sup>25].

Our approach utilizes RAG to fetch relevant Austrian federal laws, which is explained in more detail in Section 4.1. In contrast to its application as a hallucination mitigation strategy, especially for the case of missing domain-specific data, we employ RAG as a simple tool to provide the legal context for the reasoning task. In other words, RAG is a component of our approach; however, it is not explicitly used as a technique for mitigating hallucinations.

Furthermore, another method that provides external data during generation is the utilization of explicit knowledge provided by knowledge graphs (KGs). KGs incorporate a semi-structured data model that represents real-world data as triples composed of entities, visualized as nodes, and a relation, modeled as the connecting edge. The semantics of the representation are defined by an ontology, the schema that encodes the properties of a specific domain, including types of entities and relations and constraints thereof [PXNO23, BFGS19]. As such, KGs are a structured data source for augmented generations, which is referred to as GraphRAG [HWS<sup>+</sup>25], and are especially suitable to improve and verify the factuality of LLM responses, as achieved in FLEEK [BQH<sup>+</sup>23]. It is an automatic approach to extract claims from a text and to verify their factuality using a knowledge graph.

In terms of legal application, KGs are useful for understanding the relationships between legal documents, such as references, hierarchies, or rules of precedence. Specifically, for Austrian law, there exists a work that comprises a KG addressing these issues [Fil21]. As we neglect this in our setting, we do not utilize knowledge graphs and instead use plain RAG. Thereby, we avoid constructing a suitable knowledge graph that would go beyond the scope of this thesis. Nevertheless, there are applications that utilize knowledge graphs to reduce legal hallucinations [CCZ<sup>+</sup>25, Col24]. There also exist numerous approaches that combine several of these techniques to avoid hallucinations, specifically in domain-specific settings [BGW<sup>+</sup>24].

Prompt engineering approaches conclude this brief introduction to hallucination mitigation paradigms. As explained below, we conclude that our approach is a member of this class and thus, we give a more detailed comparison to prompting-based approaches and underline their difference to *Chain-of-Regulation*.

### 3.1.2 Prompt Engineering-Based Mitigation Techniques

Prompt engineering is a technique that aims to optimize the performance of an LLM in terms of desired criteria, such as mitigating hallucination, by carefully designing the prompt given to an LLM. It utilizes the model’s adaptability to a large variety of tasks

captured during the foundation training. Hence, instead of adapting a model’s parameters or providing additional infrastructure, such as vector stores in the case of RAG, the input to the model is optimized to perform specific tasks or achieve desired objectives [SSS<sup>+</sup>25]. Arguably, this is inadequate to compensate for a lack of domain-specific data, which is why our approach additionally leverages a RAG component.

According to the taxonomy in Figure 3.1, our approach is a prompt engineering technique, specifically a combination of *Self Refinement through Feedback and Reasoning* and *Prompt Tuning*. This can be justified by the fact that the intermediate regulations performed during *Chain-of-Regulation*, which are explained in more detail in Section 4.3), usually operate on subtask results. In other words, the model is prompted to improve its initial answer to a task, so it *self-refines*. Additionally, the initial task, which involves legal reasoning, is decomposed into a sequence of less complex subtasks, carefully designed and explained in Section 4.2.

A proven technique that allows LLMs to perform a lot better in specific tasks only based on a few examples is called few-shot prompting, where, motivated by the capability of humans to perform a task solely on a few examples, input-output pairs of a task are added to a prompt [ea20]. While this is a necessary pattern for base models to solve tasks, it can also be applied to instruction-tuned models, which we utilize, to enhance their performance. In our case, we apply few-shot examples for specific subtasks, and we found it especially useful when it comes to structured outputs. In comparison to our approach, few-shot prompting is a valid strategy for overcoming hallucinations in relatively simple tasks. However, it falls short in the case of the entire multi-step reasoning task, as providing few-shot examples with all associated reasoning steps is challenging. A more detailed explanation of our prompting strategies can be found in Section 4.1.

The most relevant prompting technique for this thesis is *Decomposed Prompting* (DecomP) [KTF<sup>+</sup>23]. Its concept is to decompose the initial prompt into a sequence of less complex subtasks that are solved sequentially, such that solutions are used by subsequent subtasks and the final solution is a solution to the initial task. Based on this, the authors of this approach suggest a notion of modularity of subtasks so that they can be easily replaced with more effective prompts, decomposed further, and solved by dedicated subtask handlers, such as fine-tuned models. While these properties are not utilized in our approach, DecomP also underlines the possibility to debug subtask handlers in isolation, as you have access to intermediate task results, which is the pivotal property exploited by the *Chain-of-Regulation* approach. As such, our work can be seen as an extension of DecomP as we inject intermediate regulations on the subtask results to mitigate hallucination in the multi-hop reasoning setting. In addition, we use it to observe the development of the propagation of hallucinations as described in Section 4.5. So, in contrast to the existing decomposed prompting method, we do not use specific subtask-handlers. Instead, we use subtask-specific prompts enhanced with few-shot examples on a single model. Furthermore, we regulate, i.e., improve subtask results, with the purpose of mitigating hallucinations, which is not part of the DecomP work.

Another approach that is based on decomposed prompting is Least-to-Most Prompting

[ZSH<sup>+</sup>23]. In this approach, the model first decomposes the initial task into a series of easier subproblems such that a subtask is more complex than its predecessor. Then, similarly to decomposed prompting, the subproblems are solved sequentially. The underlying concept of this method is to append the previous subproblems with their solutions to the prompt of the current subproblem, as they are intended to simplify it. Hence, the differences to our approach are essentially the same as for DecomP, as both do not apply intermediate regulations.

A popular method that employs self-refinement to reduce hallucinations is Chain-of-Verification (CoVe) [DKX<sup>+</sup>23]. In summary, CoVe consists of a first baseline response generation based on the initial prompt, which is followed by the generation of verification questions regarding this response. Then, the model answers the verification questions and uses the answers as feedback to construct a final response. As such, it can be seen as an alternative to the regulation techniques that we employ (Section 4.3). By using multiple verification questions, it achieves the decomposition of the verification task. In contrast to CoVe, we use subtask-specific prompts for regulation as well as task-agnostic prompts designed to improve answers within a single LLM call, and hence without verification questions. Furthermore, we decompose the actual task instead of the verification task.

An alternative approach, that is simple but surprisingly effective in improving the ability of large language models to perform complex tasks, especially those that involve multi-step reasoning, is Chain-of-Thought (CoT) [WWS<sup>+</sup>23]. It works by breaking down a task into a series of intermediate reasoning steps demonstrated by few-shot examples. Even though CoT achieves a decomposition of the task, its most important difference to DecomP, and thus to our approach, is that this decomposition happens model-internally because the task is still solved based on a single prompt within a single inference. As a result, the reasoning chain of CoT is returned together with the final solution within a single output, which prevents intermediate improvements. Hence, while Chain-of-Thought achieves hallucination mitigation using an internal decomposition of the task, *Chain-of-Regulation* approaches this issue by solving subtasks in subsequent inferences with intermediate result polishing.

In case there are no few-shot examples that suggest a reasonable task decomposition to the model, CoT is also applied in a zero-shot fashion, by simply prompting it to *think* step-by-step. Due to its popularity, we use zero-shot Chain-of-Thought as the main baseline for benchmarking in terms of hallucination mitigation. In addition, zero-shot CoT does not require a manually crafted task decomposition as it is implicitly handled internally by the model. Therefore, this is another point in which CoT differs from *Chain-of-Regulation*, as the manual task decomposition is an integral part of our work.

Based on this overview of existing hallucination mitigation techniques, we can identify the research gap that motivates our work. Specifically, by exposing intermediate subtask results, DecomP creates a powerful opportunity to observe and intervene in the reasoning process of language models with the purpose of mitigating hallucination, which is not yet utilized by existing approaches, including DecomP itself, as it solely transfers the raw intermediate results to subsequent subtasks without analyzing or improving them

according to an optimization objective. *Chain-of-Regulation* closes this gap by manipulating intermediate results to reduce hallucination contained in the final task answer. In summary, both task decomposition and strategies for manipulating intermediate results are already well-established in the literature. However, the combination of both is indeed novel. In addition, our approach contributes to a better understanding of hallucination in multi-hop reasoning environments by observing its development and propagation, again made possible by the access to intermediate results.

## 3.2 Hallucination Detection & Evaluation Strategies

In this section, we explore different strategies to detect hallucinations in LLM outputs, which is fundamental to evaluate *Chain-of-Regulation*. We begin by specifying the evaluation setting of our work, and based on that, we first discuss hallucination detection mechanisms in general before focusing on methods that are more suitable for our definition of hallucination. We highlight the differences from existing approaches to our evaluation method, the recursive grounder.

In contrast to a ground-truth-based setting, where hallucination is usually determined through comparative operations [RAT25], comparisons are not feasible in the case of a missing gold standard. Settings without expected output data are typically referred to as reference-free [LZB<sup>+</sup>22]. We have no legal consequences available, and hence we are in a reference-free evaluation setting.

Common approaches to determine hallucination in this case are based on self-consistency [LGL<sup>+</sup>24]. In summary, the idea is that lower variability in repeated outputs of a model indicates higher factuality. The underlying assumption is that there is a correlation between a model's self-consistency and its tendency to hallucinate [DMSH24]. While we also use a self-consistency-based strategy in the evaluation, which is presented in Chapter 5, our main evaluation tool is a grounding component. We can apply it for evaluation as we overcome the missing ground-truth by our definition of hallucination (Section 2.3.2), on which the assessment is based. Evaluation tools based on a similar notion of factuality presented in the literature are introduced below. The distinction of hallucination detection in self-consistency and inference-based methods is not standard in the literature, but it follows the two different approaches that we employ.

### 3.2.1 Self-Consistency Methods

The first concept based on self-consistency is SelfCheckGPT [MLG23], which first samples multiple responses for a given task. Then, an LLM is prompted to verify for each sentence of a sample if it is supported by the other responses. As it is a self-consistency method, it is based on the idea that an LLM is likely to generate consistent answers if it has the required knowledge. Besides the grounding component, we also employ a self-consistency method that is based on this concept. As our answers are given on a sentence level, in the form of legal consequences, we also check if they are supported by other samples. In

contrast to SelfCheckGPT, we implement the binary verification using a cosine-similarity threshold instead of an LLM as a judge. We use this embedding-based self-check due to computational restrictions, but also provide a conventional SelfCheckGPT-based cross-comparison implementation. For further details, refer to Section 5.3.

Another standard reference-free hallucination detection method is ChainPoll [FS23]. First, it asks an LLM whether the given output contains hallucination based on a prompt that demands step-by-step reasoning. Hence, ChainPoll utilizes Chain-of-Thought prompting. Then, at the end of the output, it returns a binary decision, "yes" or "no". Then, this verification prompt is repeated multiple times, and the respective answers are collected. Finally, based on the ratio of answers that conclude that the given output is hallucinated, a score can be derived. As such, ChainPoll applies self-consistency to the verification rather than the task itself, which is more resource-efficient as tasks are typically more complex than the verification step.

Even though we perform self-consistency checks, our main evaluation objective is the hallucination ratio captured by the grounding component. We expect that decompositions involving many subtasks are more likely to be inconsistent, as subtle deviations in subtask results can create varying contexts for subsequent subtasks. Therefore, comparing self-consistency among different levels of decomposition could distort the results in favor of higher-level decompositions.

### 3.2.2 Inference-based Methods

To overcome the issue of a missing ground truth, we provide a definition of hallucination, which allows a custom approach, the grounding component, to verify outputs to the legal reasoning task. Similarly, there exist detection approaches that use a similar notion of hallucination, which is based on grounding facts. Accordingly, we summarize them as inference-based methods.

Chain of Natural Language Inference (CoNLI) [LLH<sup>+</sup>23] introduces a detection agent that first extracts the claimed facts of an output to verify. Then, each fact is verified using Natural Language Inference (NLI), i.e., an LLM judges if the fact is entailed, neutral, or contradicts the provided source documents. Then, CoNLI also performs an entity-level verification to check if the entities mentioned in the claims are grounded, i.e., contained in the provided source. While the entity-level check is not relevant in the context of our work, the sentence-level NLI check offers an opportunity to judge the logical connection between a fact and a provided source text and hence is related to the base case of the recursive grounder component. It checks if a fact is contained in the scenario, the contract, or the assumptions. Specifically, in contrast to CoNLI, we classify the connection between a fact and a text into *contains*, *entails*, *contradicts*, and *independent*. In our case, a fact is grounded only in the first case, i.e., it is contained in the text. Furthermore, the recursive grounder is, as its name implies, recursive, insofar as a fact may also be grounded by an implication rule. CoNLI does not use this concept.

Fine-grained Atomic Evaluation of Factual Precision (FActScore) [MKL<sup>+</sup>23] detects hallucination in two stages. First, it decomposes a text to verify into its atomic facts, which they define as statements containing only a single piece of information. Then, they retrieve evidence from external sources (e.g., Wikipedia) based on which an LLM is prompted to verify if the evidence supports the fact. This approach allows FActScore to assign non-binary hallucination judgements on the text-level, as they return the ratio of grounded facts as a score. We also use the idea to evaluate task results based on the ratio of hallucinated facts, in our case, legal consequences. Moreover, although in the context of hallucination detection, we also utilize the notion of atomizing a text into its facts, namely, for the low-level decomposition approach (Section 4.2).

FActScore’s detection of ungrounded facts in multi-hop reasoning settings relies on the model’s capabilities to perform the reasoning internally, e.g., in a Chain-of-Thought approach, whereas our recursive grounding component supports this by design, as for each logical entailment it matches a suitable implication rule. Hence, the underlying model only needs to judge the direct logical connection between a fact and its presumably supporting text, rather than deriving the entire reasoning chain within a single inference. When evaluating the recursive grounding component, we compare it to the so-called shallow grounder, which is based on the concept behind FActScore. Based on the relevant legal text, the contract, the scenario, and the assumptions, the LLM verifies the factuality of each legal consequence within a single inference; hence, it also relies on the model’s internal reasoning capabilities. For more details on the shallow grounder, we refer to Chapter 5.

A Neuro-Symbolic Inference Engine for Grounded, Compositional, and Explainable Reasoning (NELLIE) [WCD24] is another sophisticated method to verify if a given claim is grounded. First, it generates decomposition rule candidates that support the initial claim. This allows for the recursive verification of the conditions of a decomposition rule, which are either grounded in retrieved external knowledge sources or further decomposed. The recursive character is a similarity to our evaluation component, which grounds facts either directly in the scenario or matches a suitable implication rule to verify its premises recursively. However, there are two significant differences between NELLIE and our recursive grounder. First, we do not compose decomposition rules. Instead, we search for a suitable implication rule contained in the available laws. Hence, the rules themselves are supported by external sources, in our case, Austrian law. In addition, in our definition, facts are only grounded if they are directly contained in the scenario, assumptions, or the contract. Hence, grounding facts using retrieved sources, as done by NELLIE, is not allowed in this setting.

# Methodology

This chapter describes the methodologies used to implement *Chain-of-Regulation* and its evaluation. First, we give an overview of the environmental setup that provides the infrastructure for this work. Then, we focus on the decomposition of the legal reasoning task, which we approach by developing different levels of decomposition to answer Research Question 1.1. Then, we explore different variants of LLM-driven intermediate regulation, which enable us to answer Research Question 1.2. The combination of these two subgoals constitutes the fundamental principle underlying this thesis, which is illustrated in Figure 4.1. It demonstrates our approach, which first decomposes the multi-hop legal reasoning task into  $n$  subtasks, where each subtask  $i$  is represented by a query  $q_i$  and associated with a regulation, which is directly applied to its output  $r'_i$ , resulting in  $r_i$ . The subtasks are executed sequentially, resulting in the illustrated sequence. The decomposition is manually specified while both the subtasks and the regulations are LLM-driven.

With reference to Research Question 1.3, we explain the methodology adopted to evaluate our approach in terms of hallucination mitigation. In essence, it is based on our definition of hallucination and results in the recursive grounding component. Finally, to answer Research Question 1.4, we design quantitative and qualitative methods to describe the development and propagation of hallucinations.

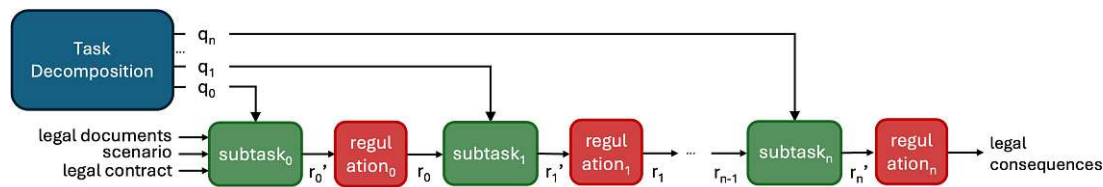


Figure 4.1: A conceptual overview of the core idea of *Chain-of-Regulation*, namely a combination of task decomposition with intermediate subtask result regulation.

Criteria	Gemma 3 27b	LLama 3.3 70b	DeepSeek-R1 32b
vRAM	17GB	43GB	20GB
context size	128k	128k	128k
throughput [tokens/sec]	100	60	60
supports German	yes	yes	yes
self-hostable	yes	yes	yes
instruction-tuned	yes	yes	yes

Table 4.1: Model comparison based on environmental constraints and required properties.

## 4.1 Environmental Setup

Before presenting the methodologies employed to answer the research questions of this thesis, we provide a detailed description of the environmental setup required to build the infrastructure on which *Chain-of-Regulation* is based. Specifically, this section addresses the model selection problem, the implementation of the RAG system, the task setup, and an overview of the prompt design strategy in use.

### 4.1.1 Constraints & Infrastructure

First, we identify important limitations of the environment of our approach, which are essential to define requirements for the LLM used as a task backend. According to these specifications, we compare different models and justify why Google’s Gemma 3 [Tea25] is the most suitable model for our use case.

The most important part of our task environment is the 54 legal contracts based on which we defined the legal reasoning task, as described in Section 2.2. They are non-disclosable, which prevents us from leveraging an LLM exposed by a public API. Hence, we need a self-hostable LLM. Furthermore, the model is required to support the German language, as both the relevant Austrian laws used and the contracts are formulated in German. We also need an instruction-tuned LLM as we do not perform fine-tuning and utilize few-shot learning only for specific subtasks. In addition, as our task context includes multiple legal documents and contracts of varying sizes, we desire a correspondingly large context size. The context size defines the maximum number of tokens a model can process in one generation. We aim for a context window with a size of at least 32k tokens.

Apart from these task-specific restrictions, we also have to cope with computational restrictions as the model is self-hosted. The infrastructure is based on TU Wien’s datalab, which offers an A100 with 80GB vRAM. Due to the limited computational resources, we strive for high throughput in terms of tokens per second. On this basis, we compare the most prominent models that best meet these requirements in Table 4.1.

We consider Gemma 3 with 27 billion parameters [Tea25], Llama 3.3 with 70 billion parameters [ea24a], and DeepSeek-R1 with 32 billion parameters [DA25], which are currently all among the most popular self-hosted LLMs. We restrict the model selection

Parameters used for Gemma 3 27b with Ollama	
context size	32k
embedding length	5376
quantization	Q4_K_M
stop	"<end_of_turn>"
temperature	1
top_k	64
top_p	0.95

Table 4.2: Important parameters for Gemma 3 27b specified by Ollama.

to these models even though we are aware that there are models fine-tuned for the German language [P13, Sch23]. However, the determination of the best model for our task is beyond the scope of this work, and thus, we restrict the model selection problem to these three popular open-source LLMs.

As can be seen from the table, all models fit into the GPU together with the K/V cache, provided that a 4-bit quantization is used. Likewise, all models share the same context size, which suffices for the required 32k tokens. Furthermore, even though Gemma 3 27b is slightly outperformed in most benchmarks by the other models <sup>1</sup>, it is designed to run efficiently on a single GPU. Therefore, it results in the best throughput, which we captured using our setup. As a result, we utilize Gemma 3 27b for both the actual approaches, including all decomposition and regulation strategies, and the evaluation, including the recursive and shallow grounding component.

Regarding the embedding model used to embed both the legal documents and the queries for retrieval, we utilize BGE-M3 [CXZ<sup>+</sup>24], a multi-lingual model that can embed long documents of up to 8192 tokens. This context size is required to cope with the comparatively long legal documents that we use. In addition, we utilize Gemini 2.5 Pro [ea25] to generate the datasets used for evaluating the grounding components and observing the development of hallucination, as explained in Chapter 5.

The model is hosted and served by Ollama <sup>2</sup>. It is a framework for managing and serving LLMs locally through a REST API. We run Gemma 3 27b as a single instance using Ollama with a context length of 32k tokens and Ollama’s default quantization, namely Q4\_K\_M. Table 4.2 summarizes the used parameters. For the implementation of both the actual approaches and the evaluations, we use Python and Jupyter Notebooks. LangChain <sup>3</sup>, an open-source framework for working with LLMs, is utilized as an abstraction layer that simplifies interaction with the model.

<sup>1</sup><https://llm-stats.com/>

<sup>2</sup><https://ollama.com/>

<sup>3</sup><https://www.langchain.com/>

### 4.1.2 RIS RAG System

Retrieval Augmented Generation (RAG), introduced in Section 3.1 as a hallucination mitigation strategy, is used in this work to provide Austrian federal laws relevant for the contracts and their associated scenarios. As such, the RAG component is not explicitly applied to reduce hallucinations. A RAG system typically consists of two stages: the indexing stage, which is responsible for fetching and processing external sources, and storing them in a vector store ready for retrieval. Then, during the retrieval, the documents that, according to the embedding model, match a given query are returned.

The indexing process is based on the OGD-RIS API <sup>4</sup>, which provides access to Austrian legal documents served as pagged JSON resources containing metadata about the documents, such as their legal type or author, along with HTML URLs to the actual legal texts. The API supports various types of legal documents; however, we primarily utilize consolidated federal laws. To avoid indexing all federal laws, we also leverage the API's capability to add search terms to the request. The following example illustrates the resulting URL schema.

```
https://data.bka.gv.at/ris/api/v2.6/Bundesrecht?
Applikation=BrKons&
FassungVom=2025-08-21&
DokumenteProSeite=OneHundred&
Suchworte='Trademark' &
Seitennummer=1
```

The communication with the API is handled by our `RIS Scraper`, which processes the API responses into a set of documents that contain the HTML content of each resource together with its title and the URL as metadata. This allows us to trace which sources the approaches of *Chain-of-Regulation* utilize. Then, the BGE-M3 model embeds the documents, specifically, into a 1024-dimensional dense vector representation. We neglect documents that exceed the maximum context size of the model, namely 8192 tokens. Finally, the documents are stored with their embeddings in a local Milvus <sup>5</sup> vector database, which is sufficient as production deployment is beyond the scope of this thesis.

After documents have been indexed, they can be retrieved, which usually occurs immediately prior to generation. To determine the most relevant documents, the user of the RAG system, in our case, exclusively the LLM, specifies a query that is embedded using the same model. Then, according to a distance metric in the vector space, in our case the Euclidean distance (L2), the documents, whose embeddings are closest to the embedding of the query, are returned as the most relevant results. Figure 4.2 illustrates the simplified interactions involved for both indexing and retrieval of our RAG system. The documents available for retrieval are determined by a prior scraping of the RIS API using a set of search terms. Then, the documents that are in fact retrieved are based on

<sup>4</sup>[https://www.data.gv.at/katalog/dataset/ris2\\_6](https://www.data.gv.at/katalog/dataset/ris2_6)

<sup>5</sup><https://milvus.io/>

Default RIS API Search Terms
Allgemeines bürgerliches Gesetzbuch
Markenschutzrecht
Unternehmensgesetzbuch
Kartellgesetz
Wettbewerbsgesetz
Urheberrechtsgesetz

Table 4.3: The set of predefined search terms used to index potentially relevant legal documents fetched from the RIS API.

the provided query. In other words, the indexing preselects potentially relevant sources among which the relevant legal documents are determined during the retrieval.

### 4.1.3 Task Setup

We define the problem used to develop and evaluate *Chain-of-Regulation* in Section 2.2. Accordingly, we have to set up the required context of the LLM, including the contract, the scenario, and the relevant legal documents. Figure 4.3 illustrates our approach to achieve this. First, starting with the contract alone, we instruct the model to generate a scenario using a zero-shot prompt. The scenario must allow legal consequences that can be derived according to both federal laws and the contract. Hence, the scenario is specified to reuse the entities mentioned in the contract, especially people and companies. Section 2.2 contains a summarized example of a generated scenario. Regarding the length of the scenario, we do not specify any requirements. This results in scenarios composed of approximately 600 tokens on average, with lengths varying between approximately 200 and 1000 tokens.

After the scenario has been generated, the model generates search terms used to fetch and index probable legal documents given the contract and the scenario. In detail, we generate three search terms in addition to a predefined set of terms, which are given in Table 4.3. Among other regulations, the *Allgemeines bürgerliches Gesetzbuch* (ABGB) contains foundational principles for legal contracts in Austria, while we find the others relevant for the two contract types, namely employment and coexistence. This hybrid approach, combining a pre-defined set of search terms with dynamically generated ones, provides two important properties. On the one hand, it allows the manual specification of important legal regulations that are universally applicable to contract-related tasks, such as the ABGB. On the other hand, the dynamic generation of search terms ensures that legal concepts applicable to only a specific scenario are also taken into account during the legal reasoning. The model is instructed to cover all relevant legal domains based on the scenario using a zero-shot prompt.

Then, we generate four RAG queries and retrieve for each query the four most relevant legal documents, again based on the contract and the scenario. The corresponding prompt

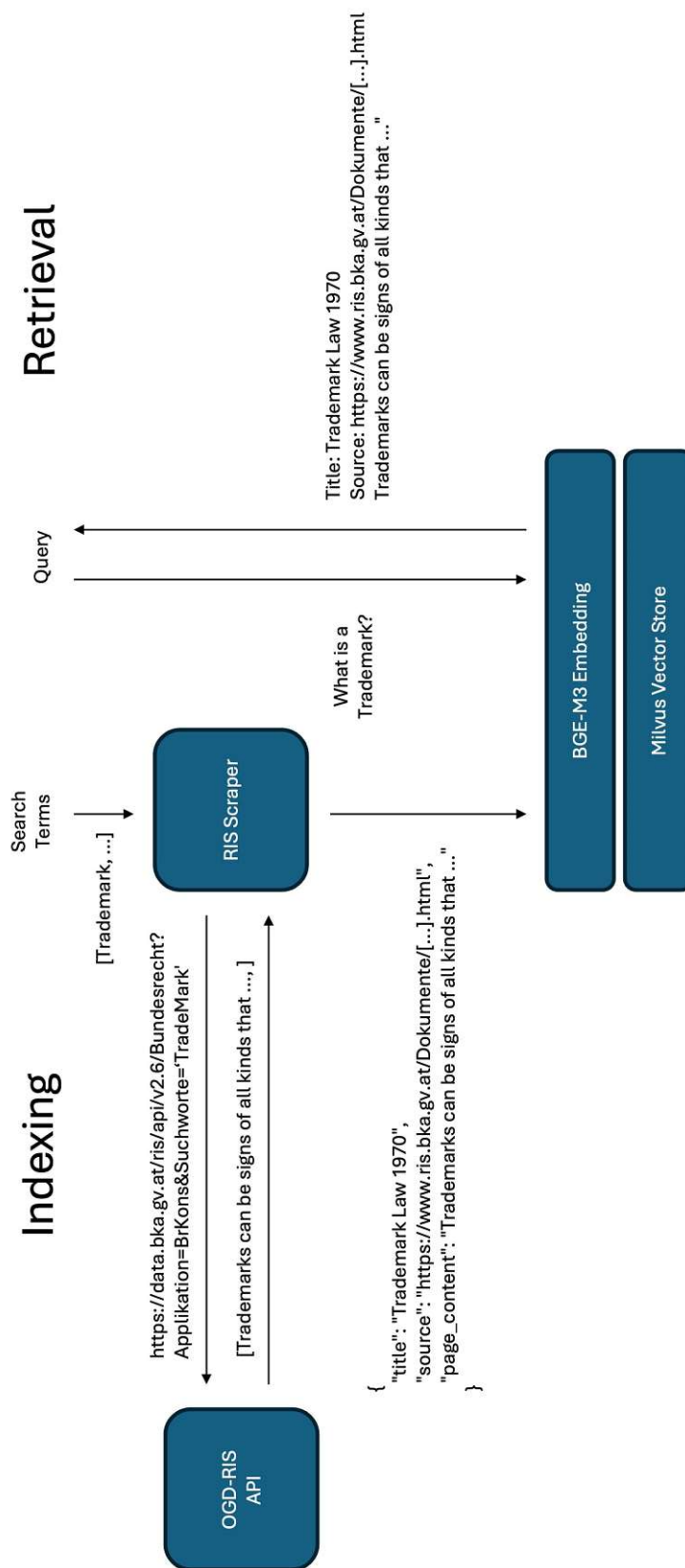


Figure 4.2: A simplified scenario that illustrates the interactions of the RAG components during indexing and retrieval.

is in a zero-shot fashion and instructs the model to neglect concrete entities from the scenario to better match the abstract formulation of the available legal documents. The number of generated queries and the number of documents per query are both parameters of the task setup. In the context of evaluation, they are used as specified above, primarily due to restricted computational resources. In general, they can be chosen arbitrarily.

Given this setup, the context is ready to use in the legal reasoning task. It is important to note that due to the fact that we compare combinations of varying decomposition levels and different regulation techniques, a consistent task setup is required. Consequently, we cache the task setup such that every approach processes the same scenario and the same set of legal documents for a given contract.

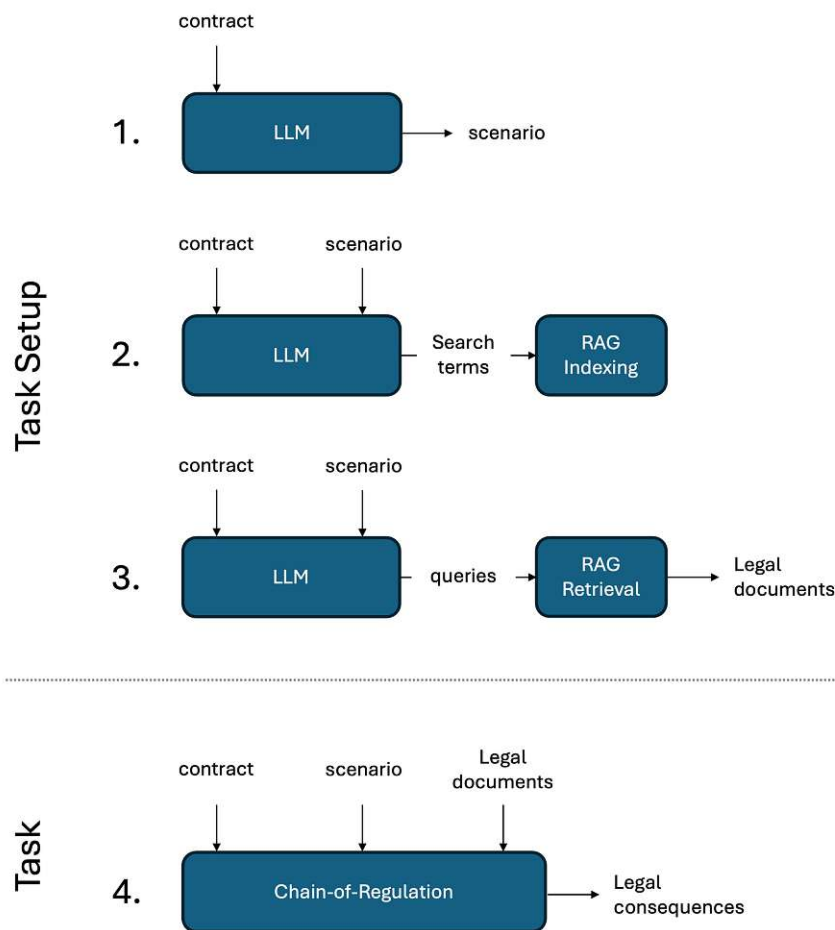


Figure 4.3: An overview of the task setup that involves the generation of a scenario, search terms, and queries to retrieve relevant legal documents. This setup is required to provide the context of *Chain-of-Regulation*.

#### 4.1.4 Prompt Design

*Chain-of-Regulation* involves the definition of many prompts, as every subtask is implemented by at least one separate prompt. As we do not utilize official prompting guidelines, we identify a few recurrent patterns during the design of our prompts. They are described below.

In general, all prompts are formulated in German to avoid deviations of prompts from the context, which is exclusively provided in German. In addition, even though the prompting concept of Gemma 3 does not distinguish between a system role and a user role, we still leverage this distinction provided by LangChain. Hence, every defined prompt consists of a system prompt, which we denote as prompt, and a user prompt, which we refer to as payload. Table 4.4 illustrates this prompt structure using a simplified scenario generation as an example. For illustration purposes, we use English prompts and fictional content.

In general, all our prompts follow to a certain extent the same structure. First, they contain an explanation of the payload, which lists all elements contained in it, such as contract, scenario, or assumptions, and a brief description of each. The payload explanation usually begins with 'The following text contains'. If required, the prompt also explains required concepts, for instance, the conceptual structure of an implication rule, before it describes the actual task to the model, typically prefixed with 'Your task is to'. Table 4.5 contains the explanations of the most important concepts.

When it comes to passing subtask results to subsequent subtasks, we observe subtle deviations from expected outputs if no further specifications on the output structure are provided. For instance, the reasoning of an implication rule application is, in addition to the actual consequence, also part of the generated output. Without further processing of this output, the reasoning is also contained in the input to the subsequent subtask. However, according to the payload description specified in the prompt, this subtask only expects the implication's consequence as input. This type of inconsistency may result in poor performance. Therefore, it is highly important that outputs which are passed as inputs to subsequent subtasks are provided in the expected form defined by their prompt.

This is analogous to the definition of interfaces when building a modular software architecture to guarantee a seamless interaction between subcomponents. In addition, interfaces allow decoupling between components and hence their exchangeability. To achieve that a subtask result conforms to a specified structure and an interface definition, fine-tuning seems to be a reasonable method. Specifically, this requires training a model to conform to a specific output structure based on a set of such outputs. However, this involves fine-tuned models for each subtask, which is not feasible due to the number of subtasks employed by *Chain-of-Regulation* in combination with our restricted computational resources.

An effective method to overcome this limitation is few-shot prompting, which we apply for those subtasks where the model's outputs for a given zero-shot prompt are not provided

---

System Message

---

###Task

The following text contains a legal contract.  
Your task is to output a specific scenario. The scenario should fulfill the following properties:

1. The scenario describes a situation to which the contract can be applied, resulting in contractual consequences.
2. The scenario describes a situation to which Austrian federal law can be applied, resulting in legal consequences.

The goal is to make the scenario legally complex, so that legal consequences in turn lead to legal consequences, and the consequences between contract and law are also intertwined. Output only the scenario, nothing else.

---

User Message

---

###Contract

Cooperation Agreement

Fruchtig GmbH (manufacturer of fruit juices, based in Linz)

Vitamina AG (manufacturer of dietary supplements, based in Innsbruck)

§ 1 - Distribution Arrangements

- (1) The parties undertake to coordinate with each other on pricing for their respective products to ensure price stability in the retail market.
- (2) The parties shall jointly define regional sales areas to avoid competitive situations.
- (3) Changes to price lists and sales territories require mutual written consent.

§ 2 - Contractual Penalty

- (1) Should either party breach the coordination agreements, it undertakes to pay a contractual penalty of €100,000 to the other party.
- 

Table 4.4: The general structure of the used prompts is defined by a system message, the actual prompt, and a user message which contains the required context as payload.

#### 4. METHODOLOGY

---

Concept	Description
Implication Rule	An implication rule consists of one or more premises and a single consequence. If all premises are met, the consequence follows. A premise is met if it is included in the scenario or the assumptions.
Fact	The following things are considered to be facts: <ol style="list-style-type: none"><li>1. Generally valid statements.</li><li>2. A definition that describes a term or concept.</li><li>3. A sentence that describes a world circumstance.</li></ol>
Assumption	It is possible to assume a fact under common sense in the context of a scenario.
Concretization	A general statement is replaced by a specific one in the context of a scenario. General expressions such as 'someone', 'a person,' etc. are replaced by specific, appropriate entities (e.g., people) of the scenario.

Table 4.5: Important concepts and a summary of their descriptions, as used in our prompts

in the expected format. Hence, in contrast to the explicit definition of interfaces as, for instance, done by using APIs, we specify interfaces implicitly using only a few examples provided to the model. An example of a prompt enhanced with few-shot examples to derive the consequence of an implication rule is given below.

**Prompt**

Your task is to apply this implication in the context of the given situation. A consequence is a concrete statement related to the scenario. Print the consequence after 'Consequence:'.

Example:

### Scenario

Max works for the bakery FooBar Ltd. and performs all his duties properly. According to the collective agreement for this industry, he is entitled to a wage of €1,800. He only receives €1,600.

### Implication

Condition: The employer pays less than the wage stipulated in the collective agreement.

Condition: The employee has performed his work properly.

Consequence: The employee can sue or demand the difference from the employer.

### Output

Consequence: Max can sue or demand the difference of €200 from FooBar Ltd.

Another important concept that we leverage is structured output, which, in the context of subtask interfaces, allows us to formalize the output format further and hence to define, e.g., hierarchical structures or lists as subtask outputs. In addition, it also facilitates the processing of concepts such as implication rules by the control program, which is the Python script that invokes the subtasks according to the decomposition. It is also responsible for passing the intermediate results to successor tasks. Furthermore, it is also used by the task decompositions to reduce the complexity of the reasoning task. Refer to Section 4.2 for more details on how structured output is utilized by decompositions.

Although LangChain offers an API for structured output, we use a custom approach that is based on the assumption that the LLM is capable of following the required output structure. Specifically, we introduce an XML-like output structure as it allows for the representation of semi-structured data that can be arbitrarily nested. So, for instance, an implication rule contains a set of premises and a single consequence, as illustrated by the example given below. Besides Implication, we use this concept to distinguish among others Fact, Assumption, Consequence, and Question elements. Except for implications, they contain only plain text.

```
<Implication>
<Condition>The employer pays less than the wage stipulated
in the collective agreement.</Condition>
<Condition>The employee has performed his work.</Condition>
<Consequence>The employee can sue or demand the difference
```

```
from the employer.</Consequence>
</Implication>
```

We use few-shot examples and what we denote as hybrid structured generation so that we can reduce the instances in which the structure is disregarded to a negligible level. This hybrid approach decomposes a task into a raw generation, which solves the actual problem, and a structured generation, which transforms the raw solution into the desired structure. The following example illustrates this two-step hybrid generation.

**Prompt**

Decompose the following text into its atomic facts.

**Payload**

Anton, who lives in Vienna, works for 'Brot&More' as a baker.

**Model Output**

Sure, here is the list of atomic facts:

- \* Anton works as a baker.
- \* Anton works for 'Brot&More'.
- \* Anton lives in Vienna.

**Prompt**

The following text contains a list of facts. Output them in a structured manner. A fact starts with <Fact> and ends with </Fact>.

**Payload**

Sure, here is the list of atomic facts:

- \* Anton works as a baker.
- \* Anton works for 'Brot&More'.
- \* Anton lives in Vienna.

**Model Output**

```
<Fact>Anton works as baker.</Fact>
<Fact>Anton works for 'Brot&More'.</Fact>
<Fact>Anton lives in Vienna.</Fact>
```

## 4.2 Task Decomposition

Based on the decomposed prompting concept [KTF<sup>+</sup>23], we modularize the legal reasoning task. This allows us to access intermediate subtask results, which are then improved to reduce hallucination, which is the core idea behind *Chain-of-Regulation*. To answer Research Question 1.1, namely, to what extent we can decompose a task for the purpose of intermediate hallucination mitigation, we introduce different levels of decomposition and compare their results in the evaluation. The decomposition is performed manually, i.e., not LLM-driven, and based on an analysis of the legal reasoning task, which is given below.

### 4.2.1 Task Analysis

As described in Section 2.2, we neglect several legal practices and reduce the application of legal norms in the context of real-world scenarios to the application of implication rules to such scenarios to derive concrete, legal consequences. Implication rules are if-then rules consisting of a set of premises that have to be met to derive the consequence. This abstraction results in a multi-hop reasoning task that is based on a real-world scenario and a set of implication rules given as raw text, in this case, a set of legal documents and a legal contract. To systematically study the decomposition of this task, we conduct a task analysis, which results in a theoretical approach to solve this task given as pseudocode in Algorithm 4.1.

First, it identifies implication rules contained in the task context. We assume that the scenario does not include implication rules. Hence, this operation is restricted to the contract  $c$  and the available Austrian federal laws  $L$ . This theoretical reasoner is designed to apply implication rules explicitly, so according to our formalization given in Section 2.3, they are extracted into a structured shape. Hence, each implication rule contains a set of premises and a consequence. Then, the reasoner determines an applicable rule, denoted by  $ir$ , i.e., a rule where all premises are met in the context. A premise is considered to be met when it is contained in the scenario  $s$ , the set of assumptions  $A$ , or the set of previous consequences  $C$  in a concrete manner. If such a rule exists, it is applied to the scenario, returning the rule's consequence concretized according to the concrete premises. In addition, this theoretical approach generates reasonable assumptions given the task context, as the task definition permits them.

Due to the multi-hop property of the reasoning, the rule selection and rule application are repeated ideally until no applicable rule remains. Furthermore, multi-hop reasoning refers to the fact that previous consequences may be concretized premises of other implication rules. Therefore, a premise is also met when it is contained in the set of previous consequences. To guarantee that this is considered by the algorithm, the theoretical multi-hop legal reasoner maintains a current state of previous consequences  $C$  and assumptions  $A$  to also check if they contain concretized premises during the application of the implication rule.

The algorithm uses the following subroutines.

- `identifyImplications( $t$ )`. This routine takes a text  $t$  and returns all its implications, i.e., the if-then rules it contains, as a set.
- `nextRule( $IR, s, c, A, C$ )`. This routine takes a set of implications  $IR$ , a scenario  $s$ , a contract  $c$ , a set of previous assumptions  $A$ , and a set of previous consequences  $C$  to return the next implication rule that can be applied. If there is no applicable rule, it returns *nil*.
- `assume( $s, c, ir, A, C$ )`. This routine derives new assumptions that are reasonable given the context, the previous assumptions, and the previous consequences.

**Algorithm 4.1:** Theoretical Multi-Hop Legal Reasoner

---

**Input:** A scenario  $s$ , a contract  $c$ , a set of legal documents  $L$   
**Output:** a set of consequences  $C$ , a set of assumptions  $A$

- 1  $C \leftarrow \{\}$
- 2  $A \leftarrow \{\}$
- 3  $IR \leftarrow \text{identifyImplications}(c) \cup \text{identifyImplications}(L)$
- 4  $ir \leftarrow \text{nextRule}(IR, s, c, A, C)$
- 5 **while**  $ir \neq \text{nil}$  **do**
- 6  $\Delta A \leftarrow \text{assume}(s, c, ir, A, C)$
- 7  $\Delta C \leftarrow \text{apply}(s, ir, A, C)$
- 8  $A \leftarrow A \cup \Delta A$
- 9  $C \leftarrow C \cup \Delta C$
- 10  $ir \leftarrow \text{nextRule}(IR, s, c, A, C)$
- 11 **end**
- 12 **return**  $A, C$

---

- $\text{apply}(s, c, ir, A, C)$ . This routine applies the implication rule  $ir$  based on the context, the previous assumptions, and the previous consequences.

The extraction of implication rules adds complexity to the task, as it requires identifying if-then rules in unstructured text and transforming them into a structured format according to the definition. Additionally, determining the next rule is complex, as it involves checking the containment of each premise in the task context until an applicable implication rule is identified. The same is true for the composition of assumptions, as contradictions to the scenario are prohibited. Then, the application of an implication rule is essentially the concretization of the rule’s consequence in the context of present, concrete premises. Hence, the combination of each subroutine and the recurring application of implication rules, caused by the multi-hop reasoning property, constitutes a highly complex legal reasoning task.

The theoretical approach guides the decomposition of the task. In particular, we implement four different levels of decomposition, which differ in how the defined subroutines are grouped in LLM-driven subtasks and how the task is split among the control program and these subtasks. We follow a top-down strategy, i.e., we start with a single prompt to solve the entire reasoning task, which involves all the identified subroutines. Then, for each decomposition level, we define subsets of the above subroutines that comprise the subtasks for a decomposition level. Specifically, the lower the decomposition level, the smaller these subsets of subroutines are.

#### 4.2.2 Monolithic Approach

The first decomposition level, which we refer to as the monolithic approach, does not use decomposition at all, i.e., the reasoning task is solved within a single LLM call. To

be more specific, according to the two-step hybrid structured generation explained in Section 4.1, it involves two LLM calls, as providing the answer in its expected structure is outsourced to a subsequent call. Apart from this, the task itself is not modularized. Hence, this implementation expects that the LLM is able to identify the implication rules contained in the legal documents, apply them in the context of the scenario, and respect the setting’s multi-hop property. Besides the actual instruction, this approach requires a detailed description of the setting contained in the prompt, especially a specification of the multi-hop character of the reasoning.

Due to its popularity in terms of hallucination mitigation, we implement the monolithic generation as a Chain-of-Thought [WWS<sup>+</sup>23] prompt, which is explained in Chapter 3. As such, the model is prompted to think *step-by-step*. However, we do not suggest solutions in the prompt, which is referred to as zero-shot Chain-of-Thought. As such, the model neither extracts rules into the structured shape suggested by our formalization nor does it utilize this structure to derive new legal consequences. Instead, the rule application is performed using natural language, which may even be easier for an LLM. Hence, we consider this approach to be an important benchmark, which is why we use this solution as the main baseline in the evaluation, such that all implementations of *Chain-of-Regulation* are compared to it.

We also apply the intermediate regulation techniques to the monolithic output for enhanced comparison during evaluation. As a result, comparing this implementation with lower-level decompositions allows us to observe the differences in a model’s internal reasoning capabilities from the manual task decomposition designed according to the task analysis. The example provided below illustrates the monolithic approach. For brevity, it neglects the hybrid structured generation and simplifies the prompt.

#### **Prompt**

The following text contains a scenario, a legal contract, and a set of implication rules. An implication consists of a set of premises and a consequence. If all premises are met, the consequence follows. Implication rules usually follow an if-then structure.

The contract may also contain implication rules.

Your task is to apply applicable implication rules to the scenario to derive all corresponding legal consequences. Dependencies between implications: Apply implications iteratively, as consequences can satisfy the premises of other implications.

Assumption: If a premise is not met, you can decide if it makes sense to assume it.

Consequence: A consequence is a concrete statement related to the scenario that follows by applying an implication.

Approach this task step-by-step.

Output the consequences and your assumptions.

```
Payload
###Contract
Cooperation Agreement ...

###Scenario
Max is currently ...

###Implication Rules
Title: Trademark Law 1970
Source: https://www.ris.bka.gv.at/Dokumente/[...].html
Trademarks can be signs of all kinds that ...

Title: Cartel Act 2005
Source: https://www.ris.bka.gv.at/Dokumente/[...].html
The abuse of a dominant market position is prohibited ...
```

### 4.2.3 High-Level Approach

We hypothesize that the recurring character of the reasoning task, imposed by the multi-hop setting, adds the most complexity because, in addition to the repetition itself, the dependencies of implication rules determine a notion of application order. To understand the meaning of dependencies between implication rules, take, for instance, an implication rule  $A$  whose consequence is the premise of another implication rule  $B$ . This requires the model to resolve this constraint and apply  $A$  before  $B$ . As we have a whole set of implication rules, determining the correct order might result in a highly sophisticated combinatorial problem.

This motivates the first level of decomposition, which we refer to as the high-level approach. It reduces the complexity of the task by outsourcing the multi-hop character of the reasoning to the control program. Hence, by decomposing the original multi-hop problem into a series of single-hop problems, we reduce the LLM's task to a single implication rule application at a time. This introduces a loop in the control program, which is modeled by pseudocode in Algorithm 4.2. Note, in comparison to Algorithm 4.1, this approach does not explicitly identify implication rules and instead operates on legal documents provided in natural language, denoted by  $L$ . As such, the model is expected to identify the implications of a legal document  $l$  during the `applyHighLevel` routine. Furthermore, the application of the implication rules is implicit according to our formalization, which means that the premises and the consequence of the rule are contained in an unstructured text. In other words, the model does not utilize structured implication rules but instead performs the reasoning using natural language. The parameter `maxIt` ensures the termination of this approach.

The algorithm uses the following two subtasks:

- `nextRule( $L, s, c, A, C$ )`. This task takes a set of legal documents  $L$ , a

**Algorithm 4.2:** High-Level Decomposition Approach

**Input:** A scenario  $s$ , a contract  $c$ , a set of legal documents  $L$ , the maximum number of iterations  $maxIt$

**Output:** a set of consequences  $C$ , a set of assumptions  $A$

```

1  $C \leftarrow \{\}$ 
2  $A \leftarrow \{\}$ 
3  $l \leftarrow \text{nextRule}(L, s, c, A, C)$ 
4  $it \leftarrow 0$ 
5 while  $l \neq nil \wedge it < maxIt$  do
6    $\Delta C, \Delta A \leftarrow \text{applyHighLevel}(s, c, l, A, C)$ 
7    $A \leftarrow A \cup \Delta A$ 
8    $C \leftarrow C \cup \Delta C$ 
9    $L \leftarrow L \setminus \{l\}$ 
10   $ir \leftarrow \text{nextRule}(L, s, c, A, C)$ 
11   $it \leftarrow it + 1$ 
12 end
13 return  $A, C$ 

```

scenario  $s$ , a contract  $c$ , a set of previous assumptions  $A$ , a set of previous consequences  $C$ , and returns the next legal document used to derive legal consequences. If there is no applicable rule, it returns  $nil$ .

- $\text{applyHighLevel}(s, c, l, A, C)$ . This task applies implication rules contained in a legal document  $l$  based on the context, the previous assumptions, and the previous consequences.

While this decomposition reduces the complexity for the language model, it introduces complexity to the control program, which now has to tackle what we refer to as the rule selection problem. Specifically, as we are in a multi-hop reasoning setting, legal consequences may satisfy premises of further implication rules. Hence, every successful rule application, i.e., one that results in at least one new consequence, may result in an update of the set of applicable implication rules. The order in which rules are applied is not significant as long as the loop terminates only if there is no applicable rule left. This guarantees that every applicable rule is eventually applied. However, given a set of implication rules, naively probing them at every iteration for applicability is too resource-consuming, as the majority of the implication rules are not applicable.

To solve this problem, the `nextRule` task is based on a rule selection prompt that instructs the model to identify a rule that can be applied in the given context and results in a new consequence. As such, this can be considered as an LLM-based heuristic to improve over the naive rule probing. In contrast to the ideal rule selection, the loop terminates after a predefined number of iterations, and implication rules are applied at most once, both for the purpose of efficiency.

In a sense, the rule selection is polymorphic, since at lower levels of decomposition it selects implication rules, represented using the proposed XML-like structure, rather than entire legal documents, provided in natural language. In addition, we restrict the number of rules among which the model has to choose and shuffle them at every iteration to avoid a biased selection. In case the LLM cannot determine a suitable rule, the program terminates. Below is an example given that demonstrates the rule selection prompt. Specifically, the rule selection task involves a set of implication rules and prompts the model to output the index of the rule that is most likely applicable. For brevity, the examples neglect few-shot examples and simplify the prompt. It is illustrated with a set of structured implication rules as used by the following decomposition levels, while the same prompt is used for the legal texts provided in natural language as used by this approach.

### **Prompt**

The following text contains a scenario and a set of implication rules. An implication rule consists of a set of premises and a consequence.

If all premises are met, the consequence follows.

Implication rules usually follow an if-then structure.

Your task is to determine the implication rule that best meets the following criteria:

- \* It can be applied to the scenario.
- \* It results in a new consequence.

Consider each implication rule separately and output the index of the best implication rule.

If none of the implications meet these criteria, just print 'no result'.

### **Payload**

###Scenario

Max is currently ...

###Implication Rules

<Implication-Start, Index: 0>

<Implication>

...

</Implication>

<Implication-End>

<Implication-Start, Index: 1>

<Implication>

...

</Implication>

<Implication-End>

In `applyHighLevel`, the model is prompted to generate legal consequences following from a provided legal document, given the task context, and to specify reasonable assumptions within one hybrid structured generation. Concretely, the model is given the legal document, usually a single legal section, and prompted to apply all contained, applicable implication rules in the context of the scenario and the contract. The following example illustrates the high-level rule application. For brevity, the prompt is simplified, and the hybrid structured generation is neglected. Note that the prompt is similar to the monolithic rule application, except for the multi-hop requirement, which is outsourced to the control program. Also, the Chain-of-Thought character is missing.

#### **Prompt**

The following text contains a scenario, a legal contract, a set of assumptions, and a text that contains implication rules. An implication rule consists of a set of premises and a consequence. If all premises are met, the consequence follows. Implication rules usually follow an if-then structure.

The contract may also contain implication rules.

Your task is to apply applicable implication rules to the scenario to derive all corresponding legal consequences.

Assumption: If a premise is not met, you can decide if it makes sense to assume it.

Consequence: A consequence is a concrete statement related to the scenario that follows by applying an implication.

Only output consequences that are not yet contained in the scenario.

Output the consequences and your assumptions.

#### **Payload**

###Assumptions

\* Max is of legal age.

\* ...

###Contract

Cooperation Agreement ...

###Scenario

Max is currently ...

###Implication Rule

Title: Trademark Law 1970

Source: [https://www.ris.bka.gv.at/Dokumente/\[...\].html](https://www.ris.bka.gv.at/Dokumente/[...].html)

Trademarks can be signs of all kinds that ...

Another important aspect that the monolithic approach does not require is state man-

agement. As consequences may satisfy the premises of other rules, which is the property that makes this setting a multi-hop environment, previous consequences and assumptions are considered during both the rule selection and the rule application. State is managed by the variables  $C$ , which contains all previous consequences, and  $A$ , to represent the set of prior assumptions. Both are part of the context for the rule selection and the rule application. Note that previous consequences are appended to the scenario.

#### 4.2.4 Medium-Level Approach

The medium-level decomposition approach is motivated by the complexity of the prompt used to solve the `applyHighLevel` task. In particular, this task requires the LLM to derive legal consequences and reasonable assumptions based on a given legal document in parallel. Furthermore, the derivation of legal consequences involves both the identification and application of implication rules contained in the text, which also needs to be resolved by the model within a single hybrid structured generation. In addition, the legal document usually contains multiple implication rules. Therefore, to further reduce the complexity of this subtask, we introduce two additional subtasks, namely `identifyImplications` and `assume`. This results in Algorithm 4.3, which illustrates this approach in pseudocode. As such, it directly follows the concept of the theoretical multi-hop reasoner, except for the parameter `maxIt`, which again allows for restricting the maximum number of iterations.

---

#### Algorithm 4.3: Medium-Level Decomposition Approach

---

**Input:** A scenario  $s$ , a contract  $c$ , a set of legal documents  $L$ , the maximum number of iterations  $maxIt$

**Output:** a set of consequences  $C$ , a set of assumptions  $A$

```

1  $C \leftarrow \{\}$ 
2  $A \leftarrow \{\}$ 
3  $IR \leftarrow \text{identifyImplications}(c) \cup \text{identifyImplications}(L)$ 
4  $ir \leftarrow \text{nextRule}(IR, s, c, A, C)$ 
5  $it \leftarrow 0$ 
6 while  $ir \neq nil \wedge it < maxIt$  do
7    $\Delta A \leftarrow \text{assume}(s, c, ir, A, C)$ 
8    $A \leftarrow A \cup \Delta A$ 
9    $\Delta C \leftarrow \text{apply}(s, ir, A, C)$ 
10   $C \leftarrow C \cup \Delta C$ 
11   $ir \leftarrow \text{nextRule}(IR, s, c, A, C)$ 
12   $it \leftarrow it + 1$ 
13 end
14 return  $A, C$ 

```

---

The `nextRule` subroutine is the same as used by the high-level decomposition approach. Apart from this, the algorithm uses the following subtasks:

- `identifyImplications(t)`. This task takes a text  $t$  and extracts contained implication rules into the proposed, structured XML-like representation and returns them as a set.
- `assume(s, c, ir, A, C)`. This task takes the task context and its state, composes reasonable assumptions based on the implication rule  $ir$  that is applied next, and returns them as a set.
- `apply(s, ir, A, C)`. This task takes the task context and its state, derives the concrete consequence of the implication rule  $ir$ , and returns it.

The identification of implication rules from the legal documents and the contract adheres to the formalization of the task setting, addressed in Section 2.3, as it extracts implication rules into a representation such that they consist of a set of premises and a single consequence. Their logical structure is as follows. If the premises are met in the context of the current state, the consequence can be derived. The current state is based on the task context, all previous consequences, and assumptions. This structured usage of implication rules implies a shift from implicit rule application to explicit rule application, as rules are only applied through the proposed structure. This also affects the rule selection. However, as addressed above, the rule selection subtask is reused as it is designed to deal with both structured and unstructured rules polymorphically. The following example illustrates a simplified implication rule extraction. In fact, this task also incorporates the two-step hybrid structured generation. The example is simplified for brevity.

#### **Prompt**

The following text contains implication rules. Your task is to identify all implication rules contained in the text. If this is not possible, simply output 'no result'.

Consider the following criteria:

1. An implication rule consists of one or more premises and a consequence. If all premises are met, the consequence follows.
2. Implication rules typically have an if-then structure, which distinguishes implications from ordinary facts or statements. Do not confuse statements with implications.
3. The conditions of an implication rule are ALWAYS connected by a logical AND.
4. If a logical OR connection occurs, duplicate the implication rule for each of the OR possibilities.
5. Formulate premises and consequences in complete sentences.

6. Make sure you do not confuse premises and consequences.
7. Avoid references to other parts of the document by resolving them. Replace references with the text sections they refer to.

### **Payload**

Title: Trademark Law 1970

Source: [https://www.ris.bka.gv.at/Dokumente/\[...\].html](https://www.ris.bka.gv.at/Dokumente/[...].html)

Trademarks can be signs of all kinds that ...

Furthermore, the composition of assumptions is outsourced into a separate subtask performed just before the rule application. The rationale for this is to create the assumptions before the actual implication rule application, as this allows the model to focus on each subtask. In addition, due to the order of the subtasks, we can ensure that the created assumptions are considered during the rule application. The following example demonstrates this subtask. It is simplified for brevity.

### **Prompt**

The following text contains a scenario, a legal contract, a set of assumptions, and a single implication rule. Your task is to decide whether it is reasonable to assume premises of the implication rule in the context of the scenario.

For each premise, choose one of the following procedures:

- \* Is it common sense to assume this premise under the given scenario? Then, output the concretized premise.
- \* Is it unlikely that the premise is satisfied? Then output 'no result'.
- \* Is the premise met in the scenario? Then output 'no result'.
- \* Is the premise contained in the assumptions? Then output 'no result'.

### **Payload**

###Contract

Cooperation Agreement ...

###Scenario

Max is currently ...

###Assumptions

\* Max is of legal age.

\* ...

###Implication Rule

```
<Implication>
...
</Implication>
```

Then, in the apply subroutine, the LLM derives the legal consequence based on the structured implication rule *ir* and the current task context. The following example illustrates this. For brevity, the prompt is simplified, and the few-shot examples are neglected. In contrast to the high-level rule application, this subtask only applies a single, explicit implication rule and hence derives at most one new consequence.

#### **Prompt**

The following text contains a scenario describing the current situation and an implication rule. Your task is to apply this implication rule in the context of the scenario to derive the consequence.

A consequence is a concrete statement in the context of the given scenario.

Print the consequence at the end of your answer after 'Consequence: '.

If the rule is not applicable, print 'not possible'.

#### **Payload**

```
###Contract
Cooperation Agreement ...
```

```
###Scenario
Max is currently ...
```

```
###Assumptions
* Max is of legal age.
* ...
```

```
###Implication Rule
<Implication>
...
</Implication>
```

### 4.2.5 Low-Level Approach

The last decomposition level, denoted as the low-level approach, builds upon the medium level and is intended to further reduce complexity by two adaptations. First, as the components of implication rules, namely the premises and the consequence, are both expressed as facts according to the formalization in 2.3, we reduce the context of the task, including the scenario and the contract, to a set of facts. Therefore, the LLM solely reasons on a set of facts and a set of implication rules to derive legal consequences, which

are also expressed as facts. According to our formalization, we denote the subtask that transforms the task context into a set of facts as `atomize`.

Secondly, during the rule application, the medium approach requires that the model extracts all concretized premises from the scenario simultaneously to derive the legal consequence. Hence, we further decompose the rule application into a sequence of premise processing subtasks and a subsequent, actual rule application. This results in Algorithm 4.4. Similar to the medium-level approach,  $IR$  is the set of structured implication rules extracted from the legal documents and the contract.  $F$  represents the set of facts extracted from the scenario and the contract.

The low-level decomposition introduces a loop over the premises of the implication rule  $ir$  and processes them separately. Specifically, `processPremise` gathers a local set  $LF$  containing context facts relevant for the rule application. Thus, during the application of the implication rule, a reduced context set  $LF$  is used by the LLM, which aims to reduce the complexity of the reasoning further. In addition, the new premise processing task generates appropriate assumptions for each premise separately. Hence, the reasoning in `applyLowLevel` is reduced to the explicit application of the implication rule  $ir$  in the context of a reduced set of facts  $LF$ . `nextRule` is the same as for previous decompositions, with the exception that the scenario and the contract are represented solely by the set of facts  $F$ .

The `nextRule` and `identifyImplications` subroutines are the same as those used by the medium-level decomposition approach. Furthermore, `applyLowLevel` is essentially the same as `apply` from the medium-level decomposition except that it uses a set of facts as task context. Apart from that, the algorithm uses the following subroutines:

- `atomize(t)`. This routine takes a text  $t$ , extracts all atomic facts that it contains, and returns them as a set.
- `processPremise(p, F, C, A)`. This routine takes a premise  $p$ , the task context, represented by a set of facts  $F$ , previous consequences  $C$ , and previous assumptions  $A$ , to reduce the entire task context to a local set of facts  $LF$ , which is used during the implication rule application. Specifically, it returns for each premise a set of assumptions and a set of facts, one of which is usually empty.

The following example demonstrates the atomization of a text into its facts. It is implemented using a hybrid structured generation, which is neglected for brevity. Note that we do not specify a definition of fact atomicity, but rather use the essential notion of atomicity, namely, indivisibility. This approach achieves acceptable results, allowing it to be implemented as a zero-shot prompt.

**Prompt**

Split the following text up into its atomic facts. Atomic means that a fact cannot be broken down any further. The following are considered facts:

**Algorithm 4.4:** Low-Level Decomposition Approach

**Input:** A scenario  $s$ , a contract  $c$ , a set of legal documents  $L$ , the maximum number of iterations  $maxIt$

**Output:** a set of consequences  $C$ , a set of assumptions  $A$

```

1  $C \leftarrow \{\}$ 
2  $A \leftarrow \{\}$ 
3  $IR \leftarrow \text{identifyImplications}(c) \cup \text{identifyImplications}(L)$ 
4  $F \leftarrow \text{atomize}(s) \cup \text{atomize}(c)$ 
5  $ir \leftarrow \text{nextRule}(IR, F, A, C)$ 
6  $it \leftarrow 0$ 
7 while  $ir \neq nil \wedge it < maxIt$  do
8    $\Delta A \leftarrow \{\}$ 
9    $LF \leftarrow \{\}$ 
10  for  $p \in ir.premises$  do
11     $dA, \Delta LF \leftarrow \text{processPremise}(p, F, C, A)$ 
12     $\Delta A \leftarrow \Delta A \cup dA$ 
13     $LF \leftarrow LF \cup \Delta LF$ 
14  end
15   $LF \leftarrow \Delta A \cup LF$ 
16   $\Delta C \leftarrow \text{applyLowLevel}(ir, LF)$ 
17   $A \leftarrow A \cup \Delta A$ 
18   $C \leftarrow C \cup \Delta C$ 
19   $ir \leftarrow \text{nextRule}(IR, F, A, C)$ 
20   $it \leftarrow it + 1$ 
21 end
22 return  $A, C$ 

```

\* Generally valid statements that are not tied to any premises.

\* A definition that describes a term or concept.

\* A sentence that describes a world circumstance.

Ensure that you list the facts thoroughly.

**Payload**

Max is of legal age and is employed by ...

`processPremise` is implemented as a hybrid structured generation that returns, for a given premise, either all facts contained in the scenario that are concretizations of it, or, in the absence of such facts, reasonable assumptions. If neither is feasible, it returns two empty sets. The following example demonstrates the underlying prompt of this subtask.

### **Prompt**

The following text contains a scenario and the premise of an implication rule. Your task is to determine whether the premise is satisfied in the context of the scenario.

Choose one of the following options:

- \* **Fact:** Is the premise contained as a concrete fact in the scenario? Then print all relevant facts.
- \* **Assumption:** Is it common sense to assume this premise under the given scenario? Then print the concretized assumption.
- \* **Discard:** Is it unlikely that the premise is satisfied? Then print 'no result'.

### **Payload**

```
### Scenario
<Fact>Max is employed.</Fact>
<Fact>Max is of legal age.</Fact>

### Premise
The employee has performed his work correctly.
```

The `applyLowLevel` subtask is also implemented as hybrid structured generation and intended to return a single legal consequence, derived from the applied implication rule. It also incorporates few-shot examples, which are neglected for demonstration purposes in the following example. In comparison to the medium-level approach, the low-level rule application uses the same prompt, but the payload is simplified by using only a local set of facts relevant to the implication rule.

### **Prompt**

The following text contains a scenario that describes the current situation and an implication rule. Your task is to apply this implication rule in the context of the scenario to derive the consequence.

A consequence is a concrete statement in the context of the given scenario.

Print the consequence at the end of your answer after 'Consequence: '.  
If the rule is not applicable, print 'not possible'.

### **Payload**

```
###Scenario
<Fact>Max is of legal age.</Fact>
<Fact>Max is employed.</Fact>
...
```

```

###Implication Rule
<Implication>
...
</Implication>

```

### 4.3 Intermediate Regulation

As discussed in Section 3.1, self-refinement prompts are a common method to reduce hallucination when applied to the model’s output for a given task. Decomposed prompting reveals subtask results, which creates the opportunity to optimize intermediate results with the purpose of mitigating hallucination in the final response. Thus, instead of improving only the final output, self-refinement prompts can be applied during the whole subtask sequence, which is presumably more effective. The verification of this hypothesis lays the foundation of this work. We refer to these subtask optimizations as regulations. In this section, we explain how we approach the development of intermediate regulation techniques to answer Research Question 1.2, which asks to what extent intermediate results can be improved so that hallucination in the final solution is reduced. Before we explain our regulation approaches, we discuss fundamental aspects.

It may be argued that instead of applying intermediate improvements, one can optimize the subtask handlers themselves, as also suggested in DecomP [KTF<sup>+</sup>23]. For instance, we can compose fine-tuned models for each subtask. However, we hypothesize that the distinction between solving the task and, based on this, refining its result is beneficial, as we achieve, in essence, a further decomposition. In other words, we argue that it is simpler to compose an initial solution and to optimize it than to generate solutions from scratch. This motivates the regulation-based approach, which requires significantly less computational resources than applying fine-tuning for every subtask.

When designing regulations of intermediate results to achieve less hallucination in the final response, we identify a significant deviation of objectives for subtask optimization and mitigating hallucination contained in the final task output. In other words, each subtask has its own notion of optimality and hence not all quality aspects of a subtask necessarily impact the factuality of the final answer. Consider, for instance, the subtask determining which implication rule to apply next. Even though an incorrect order will result in a different set of legal consequences, it does not result in hallucinated facts, provided that other subtasks are optimal. In contrast, take the extraction of an implication rule as an example, which outputs a wrong consequence. In this case, solely prompting to reduce hallucination according to our definition of hallucination is also not sufficient, as it is not applicable to the structure of an implication rule. This implies that a hypothetical regulation component focusing on our definition of hallucination is not able to identify and mitigate this error. Still, the application of this wrongly extracted rule is expected to result in a hallucinated consequence.

Based on these examples, we can identify two general approaches. On the one hand, the

subtask-specific method, which focuses on **optimizing the results of each subtask** according to its task definition. This definition is usually specified via a prompt. Hence, the first approach to resolve the above deviation is based on the underlying assumption that the better the subtask results, the better the final output in terms of hallucination mitigation. Arguably, the selection of the next rule is in fact an exception, as wrong results for almost all other subtasks, especially implication rule extraction, application, and text atomization, directly or eventually result in hallucination according to our definition. Therefore, our first approach to implement intermediate regulation is based on strategies specialized to improve the subtask results. We denote it as *task-specific* regulation.

On the other hand, the other perspective suggests a hallucination-centric approach, which prompts the model to **optimize subtask outputs with respect to reducing our notion of hallucination**. This leverages the idea of separation of concerns, as during the first LLM call, which provides the actual subtask solution, the LLM focuses on the task’s results. In contrast, during refinement, the model reduces the hallucination contained in the answer. We refer to this regulation as *hallucination-specific*.

Finally, we also define a hybrid approach, which aims to **both optimize subtask results and simultaneously reduce hallucination**, denoted as *naive* regulation. In conclusion, our approach to answer Research Question 1.2 is to implement and compare the three proposed approaches in terms of hallucination mitigation. Specifically, we perform this evaluation for each decomposition level, resulting in a set of approaches, which constitute *Chain-of-Regulation* and its baselines. A method is only considered to be part of the *Chain-of-Regulation* approaches if it in fact applies both regulation and task decomposition. Figure 4.4 illustrates the relationship between our approaches and the baselines.

### 4.3.1 Task-Specific Regulation

As motivated, the task-specific regulation approach is based on the assumption that the better the solutions to subtasks, the less hallucination is contained in the final output. In addition, it requires the decomposition to be conceptually correct, which means that if every subtask defined by the decomposition is solved without errors, the final solution also does not contain errors. In other words, this method uses local optimization, in the shape of subtask solution improvements, to mitigate the hallucination contained in the solution to the initial problem, namely, the multi-hop reasoning task.

Although there are generic strategies to improve task results, such as Chain-of-Verification [DKX<sup>+</sup>23], our task-specific regulation method defines individual approaches for every subtask, which are exclusively LLM-driven. The improvements are based on at least one additional LLM call for each subtask. In general, task-specific regulation is highly flexible and hence utilizes various concepts. For instance, a regulation may comprise a whole generation program, e.g. by using its own decomposed generation, or only a single prompt. Specifically, to give a notion of the former, the regulation of the explicit

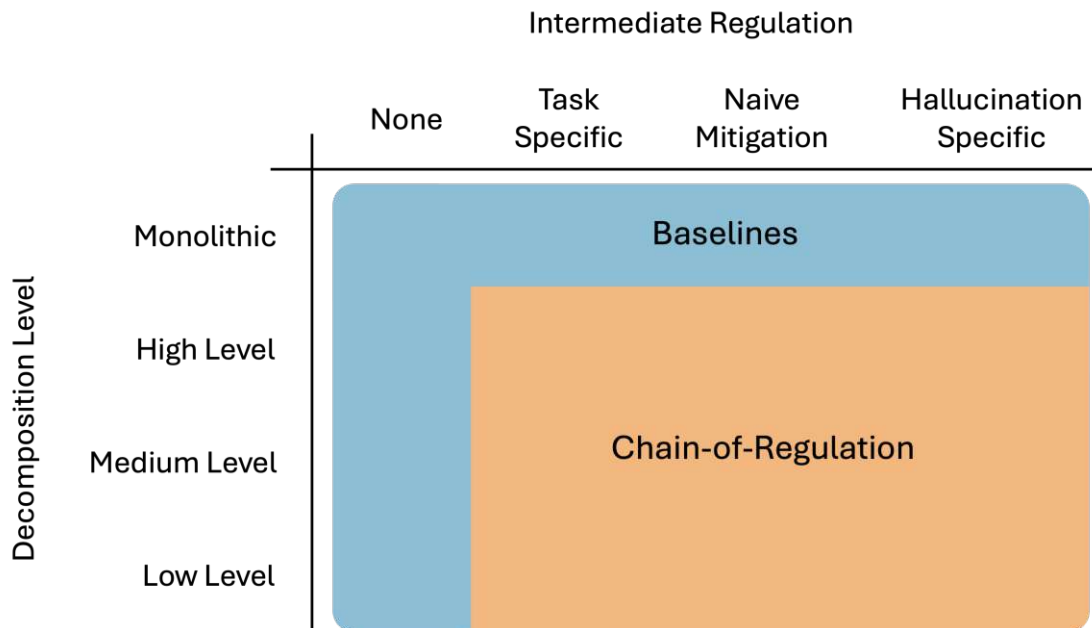


Figure 4.4: The combination of intermediate regulation and different decomposition levels results in 16 methods, which comprise our *Chain-of-Regulation* approaches and their baselines.

application of an implication rule consists of a binary classification, where the model judges and explains whether the derived consequence follows from the rule, given the task context. If this is not the case, the LLM is prompted to refine the consequence according to the given explanation. If a refinement is not feasible, the consequence is rejected. The following text provides an example of the task-specific explicit rule application regulation as used by the medium-level and low-level approaches.

**Prompt**

The following text contains an implication rule, a scenario, a contract, and a consequence that presumably follows in the context of the given scenario using the implication rule.

Your task is to assign the correct category among the following options:

1. 'correct': The consequence follows from the implication rule in the context of the scenario.
2. 'false': The consequence does not follow from the given scenario and the given implication.

Print the result and an appropriate explanation.

### **Payload**

```
###Implication Rule
<Implication>
...
</Implication>
```

```
###Scenario
Max is currently...
```

```
###Contract
Cooperation Agreement ...
```

```
###Consequence
Max can be sued because of the missing salary.
```

### **Model Output**

```
<Category>>false</Category>
<Explanation>The employer can be sued for the missing salary,
which in this case is FooBar Ltd., not Max.</Explanation>
```

### **Prompt**

The implication was incorrectly applied to the given scenario. An appropriate explanation is provided. Your task is to correct the erroneous consequence.

### **Payload**

```
###Implication Rule
<Implication>
...
</Implication>
```

```
###Scenario
Max is currently...
```

```
###Contract
Cooperation Agreement ...
```

```
###Explanation
The employer can be sued for the missing salary, which in
this case is FooBar Ltd., not Max.
```

```
###Wrong Consequence
Max can be sued because of the missing salary.
```

### **Model Output**

```
FooBar Ltd. can be sued for the missing salary.
```

Apart from that, another opportunity exploited by our task-specific regulation is to utilize the structure of the output. For instance, the extraction of implication rules from raw

text usually returns a set of rules because legal texts typically contain multiple if-then rules. As the LLM provides them according to the XML-like structure defined in Section 4.1, they can be parsed by the control program during regulation. Then, every rule is improved individually, which reduces complexity. Hence, the structure of the output is utilized to decompose the regulation task. Figure 4.5 illustrates this simple regulation program.

The regulation program begins with the implication rule extraction, which represents the actual subtask. It takes a text and returns its contained implication rules as set, denoted by  $IR$ . Then, the first regulation step aims for subtask completeness, i.e., it compares the text with  $IR$  and prompts the LLM to check if there are missing implication rules and, in this case, to output them. They are added to  $IR$ , which results in the complete set of implication rules  $IR_{Complete}$ . Then, the structure of the output is utilized by processing each implication rule  $r$  separately. The LLM decides whether the implication rule is correct given the reference text  $t$  and prints an appropriate explanation  $e$ . If this is not the case, the rule is refined based on the reference text and the explanation, resulting in the refined implication rule  $r_{Refined}$ . This is repeated for each implication rule in  $IR_{Complete}$ . As such, the regulation program aims for solution completeness and correctness.

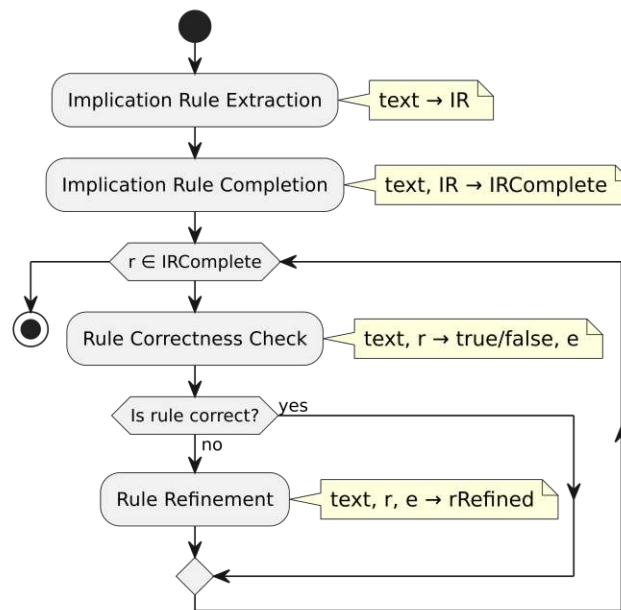


Figure 4.5: This control flow illustrates how task-specific regulation utilizes structured output to reduce the complexity of regulation tasks. In this case, this allows to separately refine structured implication rules extracted from natural language.

Furthermore, the regulation can also directly operate on the result. This concept is inspired by an approach that reduces sorting a list to pairwise ranking [ea24b]. According

to this idea, the regulation of the implication rule extraction uses the extracted rule as a payload to verify if it contains duplicated premises or if a consequence is equal to a premise, which would result in circular reasoning.

Due to its flexibility, task-specific regulation can implement almost arbitrary solution improvements, such that the inference time of the regulation may exceed that of the actual subtask. Hence, during the evaluation, we do not leverage all developed regulations to ensure scalability but instead focus on the most promising ones. This includes the implication rule extraction, implication rule application, text atomization, assumption generation, and premise processing. As we provide a task-specific regulation for every defined subtask on every decomposition level, this results in numerous task-specific regulation techniques. We summarize the most important strategies in Table 4.6. In conclusion, task-specific regulation allows the most flexible result polishing. However, it also involves the most effort, in terms of both prompt design and inference time, among the proposed regulation techniques.

Subtask	Regulation Approach
Implication Rule Extraction	First, the set of implication rules is completed by instructing the model to compare the reference text and the set of extracted implication rules, and add rules that are contained in the reference text but not in the set. Then, for every rule separately, the LLM judges if the rule is correct and prints an explanation. If this is not the case, the model is prompted to refine the rule, if feasible, based on the explanation. Otherwise, the rule is rejected.
Implication Rule Application	Based on the task context and the used implication rule, the model decides whether the generated consequence is correct and prints an explanation. If the consequence is considered to be erroneous, in a subsequent call, the LLM is prompted to refine the consequence given the task context and the previously generated explanation.
Text Atomization	Similar to the rule extraction, the first step is to complete the set of facts by comparing the reference text with the initial set of extracted facts. Then, the set of facts is passed to the model, which is instructed to remove facts not contained in the reference text. In contrast to the rule extraction, we do not split the set of facts according to the output structure, but instead refine all facts within one inference for efficiency reasons.
Premise Processing	This regulation takes the concretizations of a premise to verify if the premise is satisfied in the context of the scenario. If this is the case, the model is prompted to check whether the provided concretizations are correct and to refine them if necessary.

Table 4.6: This table summarizes the subtask-specific regulation methods used to improve the results of the most important subtasks.

### 4.3.2 Hallucination-Specific Regulation

In contrast to task-specific regulation, this approach addresses the previously identified discrepancy between subtask optimization and hallucination mitigation by prioritizing the latter over the former, aiming for reduced hallucination in intermediate outputs. However, as already demonstrated with the implication rule extraction, this approach is not capable of identifying and reducing erroneous subtask results that do not directly but eventually result in hallucination. Therefore, the underlying assumption for this approach is that the model is capable of understanding that specific errors of subtask results will ultimately result in hallucinated facts solely based on the definition of it contained in the prompt. Regarding the previous example where an implication rule is extracted with a wrong consequence, this assumption requires the model to reason that the application of a wrong implication rule would result in a hallucinated fact, and hence it needs to refine the rule.

In comparison to the previous regulation method, the hallucination-specific technique is task-agnostic and hence always employs the same strategy. Specifically, based on a universal prompt, i.e., it is used for every subtask, the LLM is required to improve the initial subtask result, which is provided in the payload. In addition, to respect the task to be regulated, the payload also contains the original prompt and the original payload of the subtask. The example provided below illustrates the hallucination-specific regulation component.

#### **Prompt**

The following text contains an original task, the original input, and your result. Your task is to minimize incorrect information in the result.

A piece of information is correct if any of the following holds:

1. The information is directly contained in the scenario.
2. The information is contained in the assumptions.  
This is restricted to tasks with inputs containing assumptions.
3. The information follows as a consequence of an applicable implication rule contained in the input. An implication is applicable if all its premises are met in the context of the scenario.

Print the improved result.

#### **Payload**

### Original Task ###

...

### Original Payload ###

...

```
### Your Result ###  
...
```

### 4.3.3 Naive Regulation

The third strategy to improve subtask results is what we refer to as naive regulation. It can be considered as a combination of the earlier approaches, as it prompts the model to improve the result of the subtask and to reduce hallucination contained in the original solution to a subtask. In particular, we neither specify hallucination nor how to improve subtasks, which is why we denote this regulation as naive. Therefore, the model is expected to improve arbitrary task results, based on the assumption that a naive decomposition into result composition and result refinement, as argued above, actually enhances the result. Additionally, this method relies on the assumption that the model possesses an internal notion of hallucination. In essence, this regulation aims to find a trade-off to deal with the aforementioned deviation between subtask optimization and hallucination reduction.

Similarly to the hallucination-specific regulation, the naive regulation is also task-agnostic, i.e., the same prompt is applied to every subtask. As such, it also employs the same payload structure containing the original prompt, the original payload, and the model's initial answer. Hence, in contrast to the task-specific approach, this significantly reduces the effort required to specify regulation prompts. This type of regulation is illustrated by the example given below.

#### **Prompt**

The following text contains an original task, the input, and the result you created.

Your task consists of two parts:

1. Improve the result with regard to the original task.
2. Improve the result with regard to hallucination. Hence, avoid uncertain or fabricated facts.

Print the improved result.

#### **Payload**

```
### Original Task ###
```

```
...
```

```
### Original Payload ###
```

```
...
```

```
### Your Result ###
```

```
...
```

## 4.4 Hallucination Detection

Section 4.2 deals with the decomposition of the legal reasoning task, and Section 4.3 explains our approach to implement intermediate regulations with the purpose of mitigating the hallucination in the final task result. Their combination is the main idea behind *Chain-of-Regulation* and thus essentially gives an answer to our research question. However, as discussed earlier, we are in a reference-free setting, i.e., there is no ground-truth data for evaluating this approach, and also no applicable, universally accepted definition of hallucination. Hence, we define hallucination in the context of formalizing our setting in Section 2.3.2. According to this definition, we also propose an idealized recursive grounder to detect hallucinated facts. This algorithm forms the foundation of the recursive grounder, which is the primary method used for evaluating *Chain-of-Regulation*. As such, it is used to quantify the effectiveness of our approach in terms of hallucination mitigation, and therefore, it is also necessary to provide a quantifiable answer to the research question of this thesis. In addition to the recursive grounder, we also introduce a monolithic evaluation approach, which we denote as the shallow grounder, and a self-consistency method based on repeated task results.

### 4.4.1 Recursive Grounder

We define the recursive grounder in Algorithm 4.5, which is based on the definition of Algorithm 2.1. In addition to the task context and the fact to ground  $f$ , this implementation uses two working variables, namely  $G$ , which contains all already grounded facts, and  $d$ , which represents the current grounding depth. Hence,  $d$  indicates the number of hops that are required by the longest reasoning chain to presumably conclude  $f$ . To guarantee termination in practice, we specify a max depth by  $d_{max}$ . The task context includes the scenario  $s$  with the contract, the set of assumptions  $a$ , and the set of implication rules  $IR$ .

To avoid repeating the grounding process for facts that have already been grounded, we use the set variable  $G$ , which contains already verified facts. The grounding procedure starts by checking if the fact is already contained in the set  $G$ . This check is LLM-driven. Accordingly, we keep  $G$  up to date and add  $f$  to  $G$  once it is verified. Then, according to the idealized algorithm, we check if  $f$  is contained in the scenario or the assumptions.

If none of the base cases is fulfilled, i.e., the fact is neither contained in the scenario nor the assumptions, the matching implication rule is determined. The algorithm illustrates the exact process of rule selection. We iterate over each implication rule and apply an LLM-driven binary decision procedure to every rule separately to verify its applicability. In practice, the binary decision is overly selective, requiring repeated iterations over the set of implication rules, as indicated by the wrapping *while* loop. Hence, besides  $d_{max}$ ,  $maxIt_{ir}$  specifies the number of decision loops and thus is the second parameter

of the grounder. This procedure implements a greedy rule determination strategy, i.e., if a matching rule is found, all other rules are disregarded. We assume that there are only rare cases of colliding implication rules, which justifies this first-come, first-served policy. In case there is no applicable implication rule, the fact is considered to be hallucinated. Otherwise, the algorithm iterates over the rule's premises, concretizes them, and recursively verifies their factuality.

The algorithm utilizes the following subroutines:

- `contains( $f$ ,  $t$ )`. This LLM-driven procedure takes a fact  $f$  and a text  $t$  and returns true if and only if the fact is contained in the text. Alternatively,  $t$  can also be a set of facts.
- `isConsequenceOf( $f$ ,  $ir$ )`. This procedure is LLM-driven, takes a fact  $f$  and an implication rule  $ir$ , and returns true if and only if the application of  $ir$  can result in  $f$ .
- `concretize( $p$ ,  $ir$ ,  $f$ ,  $s$ )`. Based on the presumable consequence  $f$ , the rule  $ir$ , and the task context, the model returns the concretized premise  $p$ .

The subroutines employed by the algorithm are each implemented by a single prompt. Specifically, under the hood, the `contains` function is not a binary classification, but instead determines the correct entailment relation between a text and a fact among four options according to our formalization in Section 2.3. A fact is considered to be grounded in the reference text only if the entailment is classified as *contained*. The instance below demonstrates the underlying prompt with an example payload. The prompt incorporates few-shot prompting and the hybrid structured generation, which are both neglected for brevity.

**Prompt**

The following text contains a reference text and a fact.  
Your task is to output the appropriate category and a reason.

###Categories

- \* 'contain': The fact is directly contained in the reference text.
- \* 'implies': The fact follows logically from the reference text.
- \* 'contradicts': The fact contradicts the reference text.
- \* 'independent': The fact has no connection to the reference text.

Here are the precedence rules to be applied when multiple categories are appropriate:

**Algorithm 4.5:** Recursive Grounder *recGround*


---

**Input:** A scenario  $s$ , a text containing all assumptions  $a$ , a set of implication rules  $IR$ , a fact  $f$  to ground, the current depth  $d$  starting with 0, the set of grounded facts  $G$  starting with an empty set

**Output:** *true* or *false*

```

1 if  $d_{max} \leq d$  then
2   | return false
3 end
4 if  $contains(f, G)$  then
5   | return true
6 end
7 if  $contains(f, s)$  then
8   |  $G \leftarrow G \cup \{f\}$ 
9   | return true
10 end
11 if  $contains(f, a)$  then
12   |  $G \leftarrow G \cup \{f\}$ 
13   | return true
14 end
15  $ir \leftarrow nil$ 
16  $it \leftarrow 0$ 
17 while  $it < maxIt_{ir} \wedge ir = nil$  do
18   | for  $ir_c \in IR$  do
19     | if  $isConsequenceOf(f, ir_c)$  then
20       |  $ir \leftarrow ir_c$ 
21       | break
22     | end
23   | end
24   |  $it \leftarrow it + 1$ 
25 end
26 if  $ir = nil$  then
27   | return false
28 end
29 for  $p \in ir.premises$  do
30   |  $p_c \leftarrow concretize(p, ir, f, s)$ 
31   | if  $\neg recGround(s, a, IR, p_c, d + 1, G)$  then
32     | return false
33   | end
34 end
35  $G \leftarrow G \cup \{f\}$ 
36 return true

```

---

```
* 'contain' takes precedence over 'implies'
* 'contradicts' takes precedence over 'independent'
* 'independent' takes precedence over 'implies'

Print the result and an explanation.

Payload
### Reference text
Max is currently ...

### Fact
Max is of legal age.
```

Furthermore, `isConsequenceOf` is represented by a prompt that instructs the model to reach a binary decision on whether the fact to ground is a concretized consequence of the implication rule. It is enhanced with few-shot examples and is responsible for matching a fact to the implication rule that was used to derive it. In the `concretize` routine, the LLM has to determine the concretized premises required to derive a fact by applying an implication rule in the context of the given scenario. To achieve this, it is necessary to identify the relations between the abstract entities contained in the premises and the consequence of the rule. Then, the abstract entities of the rule's consequence are matched to the entities of the fact. Finally, according to the entity relations within the implication rule, the concrete entities of the consequence are transferred to the abstract premises to concretize them. Figure 4.6 illustrates this.

The recursive grounder is not only able to identify hallucination as specified by our definition, but it also allows us to backtrack the derivation of consequences, which might involve several reasoning hops in our setting. Specifically, based on its recursive character, we can record the expansion of implication rules and their concretized premises. This results in what we refer to as a grounding tree. Starting from the initial fact to ground, which constitutes the root of the tree, implication rules and concretized premises are traversed as intermediate nodes until premises are expanded, which are directly grounded in the scenario or the assumptions, represented by the grounding tree's leaf nodes.

Even though a grounding tree might not reflect the actual reasoning employed by a multi-hop reasoner, especially in cases where multiple reasoning chains result in the same consequences, it allows for visually representing and retracing a valid reasoning required to derive a given fact, provided that this fact is not hallucinated. This contributes to better explainability of results and hence enables manual reviews of the LLM's reasoning process. An essential property of the recursive grounder is that it does not require access to either intermediate subtask results or internal reasoning chains, as, e.g., provided by Chain-of-Thought [WWS<sup>+</sup>23]. It works solely based on the consequence to verify and the initial task context, which is used to extract implication rules, match them with concretized facts, and, eventually, to ground facts. Illustrations of grounding trees are provided in Section 6.1. As the grounding component is our approach towards Research

## 1. Identification of abstract entity relations

### Consequence

c: Fabian can sue or demand the difference from the FooBar Ltd.

### Implication Rule

p<sub>1</sub>: **The employer** pays less than the agreed wage.

p<sub>2</sub>: **The employee** performs his/her work properly.

c: **The employee** can sue or demand the difference from **the employer**.

## 2. Matching abstract and concrete entities

### Consequence

c: **Fabian** can sue or demand the difference from the **FooBar Ltd.**

### Implication Rule

p<sub>1</sub>: FooBar Ltd. pays less than the agreed wage.

p<sub>2</sub>: Fabian performs his/her work properly.

c: **The employee** can sue or demand the difference from the **the employer**.

## 3. Concretization of premises using entity relations

### Consequence

c: Fabian can sue or demand the difference from the FooBar Ltd.

### Implication Rule

p<sub>1</sub>: **FooBar Ltd.** pays less than the agreed wage.

p<sub>2</sub>: **Fabian** performs his/her work properly.

c: **Fabian** can sue or demand the difference from the **FooBar Ltd.**

Figure 4.6: The concretization of the premises of an implication rule involves three conceptual steps, which are illustrated in this example using two concrete entities.

Question 1.3, it also requires an evaluation, which is discussed in Section 5.1. In essence, we evaluate it by comparing it to a baseline implementation, the shallow grounder.

### 4.4.2 Shallow Grounder

This alternative approach to verify the factuality of derived legal consequences is the main baseline used in the evaluation of the recursive grounder. In contrast to the former method, this strategy is more lightweight, in terms of both inference time and construction, and implemented by a single hybrid structured generation. It performs a binary classification to decide whether a fact is grounded or not. To achieve this, the prompt includes a description of when a fact is considered to be factual according to our

definition of hallucination. In addition, it explicitly states the multi-hop characteristic of the task setting to achieve a Chain-of-Thought flavored verification. As such, this approach relies on the model’s internal multi-hop reasoning capability. In general, the method is related to FActScore [MKL<sup>+</sup>23], which verifies the atomized facts of an output to assign a score to it according to the ratio of hallucinated facts. In contrast to FActScore, the results of the legal reasoning task, the legal consequences, are provided individually by design such that no atomization is required, and in addition, the shallow grounder verifies facts solely based on the task context, i.e., without dynamically retrieved sources as done by FActScore. The following example illustrates the underlying LLM prompt of the shallow grounder in a simplified manner. It is implemented in a zero-shot fashion.

### **Prompt**

The following text contains a scenario, a legal contract, a set of assumptions, a set of implications, and a fact.

A fact is verified if one of the following is true:

1. The fact is directly and meaningfully contained in the given scenario.
2. The fact is directly and meaningfully contained in the contract.
3. The fact is directly and meaningfully contained in the assumptions.
4. The fact results from the correct application of an implication.

The application of an implication is correct if all conditions are met in the context of the scenario and the consequence is derived in the context of the met conditions. Recursive Implications: Implication rules can be applied recursively in a multi-hop reasoning manner.

Answer the following question: Is the given fact verified?

### **Payload**

```
###Contract
Cooperation Agreement ...

###Scenario
Max is currently ...

###Assumptions
* Max is of legal age.
* ...

###Implication Rules
Title: Trademark Law 1970
```

```

Source: https://www.ris.bka.gv.at/Dokumente/[...].html
Trademarks can be signs of all kinds that ...
...

###Fact
Max has ...

```

### 4.4.3 Cross-Comparisons (Self-Consistency)

Apart from the hallucination detection mechanisms that rely on our definition of hallucination, we also provide a verification approach that is based on self-consistency. As introduced in 3.2, this is a common strategy employed in cases where the ground-truth is missing to evaluate LLMs in terms of hallucination. The underlying assumption is that an LLM is likely to generate consistent answers if it is confident about them. In this context, we assume that long sequences of subtasks are more likely to be inconsistent because deviations in subtask results cause varying contexts for subsequent subtasks. Therefore, according to this hypothesis, an evaluation based on self-consistency would distort the result in favor of shorter subtask chains, i.e., higher-level decompositions. As a result, we use the recursive grounder as the source of truth when it comes to the evaluation of hallucination mitigation. However, to verify how far self-consistency is in fact affected by the length of subtask chains, we perform what we denote as cross-comparisons.

Specifically, we perform the task several times, e.g., three or five times, in the same context, and then for each solution, we verify for each of its legal consequences separately if it is contained in the other solutions. Then, for each legal consequence, we determine the proportion of those solutions that contain it. The average ratio over all the legal consequences of a solution is its final cross-comparison score. This method is related to the approach suggested by SelfCheckGPT [MLG23]. To verify if a fact is contained in another solution to the task, i.e., a set of legal consequences, we use the same subroutine employed by the recursive grounder, namely `contains`. As this strategy involves many containment checks, during evaluation, we replace the LLM-based `contains` routine with an embedding-based method for efficiency reasons. In particular, for one cross-comparison, we embed one legal consequence of an answer and all the legal consequences of another answer. Then, the cosine similarity score to the closest consequence determines the self-consistency value for the given consequence and the comparing answer. Again, we average over all values to obtain the final cross-comparison score. Algorithm 4.6 illustrates how the cross-comparison score of a specific *Chain-of-Regulation* variant, i.e., a combination of a decomposition level and a regulation technique, is determined.

## 4.5 Hallucination Development & Propagation

The final subgoal of this thesis, as expressed in Research Question 1.4, aims to determine the means that allow for describing the propagation and development of hallucination in the multi-hop legal reasoning setting. As such, it contributes to a better understanding of

**Algorithm 4.6:** Embedding-Based Cross Comparison

---

**Input:** A scenario  $s$ , a legal contract  $c$ , number of cross comparisons  $c_{num}$   
**Output:** a cross comparison score  $score_{cc}$

- 1  $LCS \leftarrow \{\}$
- 2  $it \leftarrow 0$
- 3 **while**  $it < c_{num}$  **do**
- 4      $LC \leftarrow \text{ChainOfRegulation}(s, c)$
- 5      $LCS \leftarrow LCS \cup \{LC\}$
- 6      $it \leftarrow it + 1$
- 7 **end**
- 8  $Scores \leftarrow \{\}$
- 9 **for**  $LC \in LCS$  **do**
- 10     $OLCS \leftarrow LCS \setminus \{LC\}$
- 11    **for**  $lc \in LC$  **do**
- 12     **for**  $OLC \in OLCS$  **do**
- 13       $score_{lc} \leftarrow 0$
- 14      **for**  $olc \in OLCS$  **do**
- 15        **if**  $\text{cosSim}(lc, olc) > score_{lc}$  **then**
- 16           $score_{lc} \leftarrow \text{cosSim}(lc, olc)$
- 17        **end**
- 18      **end**
- 19       $Scores \leftarrow Scores \cup \{score_{lc}\}$
- 20    **end**
- 21 **end**
- 22 **end**
- 23  $score_{cc} \leftarrow \text{avg}(Scores)$
- 24 **return**  $score_{cc}$

---

how hallucination emerges, which in turn helps to better detect and avoid hallucination in the future. To achieve this, we distinguish between quantitative and qualitative perspectives. Hallucination development refers to the quantitative aspect, which aims to determine how the proportion of hallucinated facts increases with the length of reasoning chains. In contrast, hallucination propagation denotes a qualitative aspect, with the purpose of analyzing which errors occur and how they affect subsequent results. Both strategies are applied only on the low-level decomposition approach without intermediate regulation to induce more hallucination. In addition, the analysis is conducted in a ground-truth-based environment. More details on the evaluation setting can be found in Section 5.4. Section 6.3 addresses the findings of both aspects.

### 4.5.1 Quantity: Hallucination Development

We define a fact to be associated with a reasoning level  $n$  if  $n$  is the number of implication rule applications in the longest reasoning chain required to derive the fact. In other words, the reasoning level of a fact is equal to the height of the corresponding grounding tree for this fact. Intuitively, the higher the reasoning level of a consequence, the more difficult it is to obtain. We define the reasoning level of a fact directly grounded in the scenario to be 0. To determine how hallucination develops in the multi-hop legal reasoning environment, we assign a quantitative hallucination score to each reasoning level. The hallucination score of a reasoning level is the ratio of hallucinated facts among all facts that are associated with this level.

Our naive expectation is that the hallucination ratio increases the higher the reasoning level, as high-level consequences are dependent on lower reasoning levels and hence may suffer from consequential errors. This methodology allows for the numerical expression and plotting of how hallucination develops over the reasoning level. While this perspective focuses on the quantity of hallucination, the following aspect aims to analyze these consequential errors and their effects in more detail.

### 4.5.2 Quality: Hallucination Propagation

Hallucination propagation refers to a qualitative analysis of hallucination in our context. Essentially, the aim is to identify consequential errors and to describe how they affect subsequent reasoning. This is motivated by a striving to better understand these errors to mitigate them further. In fact, specific intermediate task-specific regulation approaches are inspired by findings of this analysis. We approach the observation of hallucination propagation by first identifying which subtasks may contribute to hallucination in the final task result. Based on this analysis, we determine an appropriate representation of the LLM’s reasoning process, which enables the detection of errors that lead to hallucination. This is motivated by the observation that, as discussed in Section 4.3, not all subtasks, such as the rule selection task, are relevant for hallucination in the final task result.

The analysis of subtasks prone to hallucination is restricted to the tasks of the low-level decomposition for which this analysis is conducted. First, we identify the premise processing task as the first potential error source. It is responsible for concretizing entities, which may result in a mix-up of entities or concretizations that are not even present in the scenario, leading to the incorrect application of implication rules or the propagation of incorrect entities. Both scenarios result in ungrounded consequences. Furthermore, the application of an implication rule might result in hallucination if it is applied even though not all premises are satisfied. In addition, a wrongly concretized consequence also leads to ungrounded facts. Apart from that, the extraction of implication rules also causes hallucination if performed improperly, as discussed above.

According to this analysis of relevant subtasks, we utilize what we refer to as reasoning trees, a visual representation of the reasoning process used by the model to solve the multi-hop reasoning task. Based on this visualization, we can identify errors of the

relevant subtasks prone to hallucination. Conceptually, the reasoning trees are similar to the grounding trees. Specifically, they contain facts, either directly included in the scenario or legal consequences, and implication rules, which have been applied during the reasoning, both represented by nodes. An edge in the reasoning tree is directed from a fact to an implication rule in case the fact is a required premise of the rule. In contrast, a directed edge from an implication rule to a fact indicates that the fact is the consequence of the rule.

Then, based on the reasoning tree representation, we conduct a manual analysis to verify which of the potential errors that we addressed above are significant in practice. In addition, the reasoning paths allow tracing how errors propagate to subsequent rule applications and hence hallucinated consequences. The results of this analysis of hallucination propagation enable the application of intermediate mitigation specifically targeted at prevalent errors. For detailed results on the qualitative hallucination analysis, which contain both reasoning trees and a list of revealed hallucination provoking errors, refer to Section 6.3.

# Evaluation

In this chapter, we explain how we evaluate the employed methodologies to give well-founded answers to the research questions of this thesis, which are defined in Chapter 1. In general, we are in a reference-free setting, and the underlying method for evaluating both hallucination detection and mitigation involves comparing our approaches with existing ones. Specifically, in the case of the hallucination mitigation evaluation, we compare our *Chain-of-Regulation* to a zero-shot Chain-of-Thought approach. For the hallucination detection, we compare the recursive grounder to an alternative implementation, namely, the shallow grounder. Furthermore, we also address how to approach the determination of hallucination development, which is evaluated in a ground-truth setting.

## 5.1 Evaluating the Grounder

In terms of hallucination mitigation, the recursive grounder is the primary evaluation method to assess *Chain-of-Regulation*. As such, it addresses Research Question 1.3, which asks to what degree we can detect and also quantify hallucination. To give a well-founded answer to this question, we are required to evaluate the grounder itself. Furthermore, our definition of hallucination fundamentally restricts the notion of non-factualities. Hence, instead of comparing the recursive grounder to an existing hallucination detection technique, our main baseline is the shallow grounder. It utilizes an LLM that is aware of this hallucination definition to judge the factuality of a given fact within a single Chain-of-Thought-based generation.

In general, we evaluate the grounders as binary classifiers, which distinguish factual facts, considered as the positive class, from hallucinated ones, which represent the negative class. In addition to overall accuracy, the focus of this evaluation is the false positive rate, as it indicates the classifier’s vulnerability to evaluate hallucinated facts as factual. In other words, the lower the false positive rate, the fewer hallucinations remain undetected.

ground-truth size	105	113	85	26	1
reasoning level	1	2	3	4	5

Table 5.1: Number of correct consequences for each reasoning level in the grounder evaluation dataset.

However, at the same time, we aim for a low false negative rate to prevent the detector from being overly conservative.

When it comes to evaluating the grounding components, we require a ground-truth to assess their ability to distinguish factual facts from hallucinated ones. In this case, the result is a dataset composed of facts that are correctly labeled as either hallucinated or factual. To achieve this, we synthesize a dataset with Gemini 2.5 Pro using a one-shot example, i.e., a single data sample. Essentially, each data sample comprises a scenario, a set of implication rules, a set of factual consequences, and a set of hallucinated facts. A simplified example is given below.

```

1  {
2    "scenario": "Max is currently...",
3    "correct_consequences": [
4      "Max may be prosecuted.",
5      "There is a concrete threat to public health."
6    ],
7    "wrong_consequences": [
8      "Max loses ownership of his property.",
9      "Martin may be prosecuted.",
10   ],
11   "implication_rules": [
12     "<Implication>...</Implication>"
13   ]
14 }

```

The synthetic dataset contains 630 facts, of which 330 are factual and 300 are hallucinated. Hence, the facts are approximately equally balanced among the classes. Also, we aim for a balanced distribution of facts in terms of reasoning level, which is essentially the case for reasoning levels 1 to 3. In contrast, the others are underrepresented as summarized in Table 5.1. Although we specify a maximum reasoning level of 5 during dataset generation, the LLM is only able to generate a single fact at this level. The increasing complexity of such data samples, which involve multiple implication rule interdependencies, might explain this. For the evaluation of the grounders, every fact is classified separately using its context, i.e., the implication rules and the scenario of the data sample in which it is contained.

Apart from this evaluation as binary classifiers, we also perform what we denote as consistency evaluation. Both the recursive and the shallow grounder utilize LLMs and hence are non-deterministic methods, so we are required to verify their result consistency when applied multiple times for the same input. In other words, if polled numerous times, the grounders are expected to return the same answer. To verify this property, we repeat the grounding ten times for the first 30 samples of the dataset, which contain 171 facts, of which 81 are correct and 90 are hallucinated. Then, the consistency score of a fact is the proportion of the majority label among the ten repetitions. Therefore, a score of 0.5 refers to a random binary classifier, whereas a score of 1 indicates a deterministic classification. The results of the grounding performance evaluation and its consistency test are presented in Section 6.1.

## 5.2 Trade-Off between Hallucination Mitigation & Relevance

The fundamental aim of *Chain-of-Regulation* is to minimize the proportion of ungrounded legal consequences returned as a solution to what we refer to as the multi-hop legal reasoning task. In fact, this single objective can be easily solved by a naive, alternative approach, which restricts its solution to trivial consequences for which the model is confident. Consider, for instance, a solution approach to the multi-hop legal reasoning task, which is solely based on the atomization subtask. Specifically, it atomizes the scenario and returns the set of atomic facts as a solution. According to our definition of hallucination, this solution is perfect in terms of factuality, provided that the atomization is optimal. However, the provided consequences are useless for the user, as they do not carry new information. Instead, we aim for complex, non-evident consequences, which contain new information. We refer to such facts as relevant. Therefore, to take the requirement for relevant consequences into account, we additionally evaluate our approaches with respect to relevance. This allows us to verify if they neglect relevance in favor of factuality, which is what we denote as a trade-off between hallucination mitigation and result relevance.

To assess the relevance of solutions to the legal reasoning task, we score each consequence separately and accumulate the relevance scores. As such, the more consequences a solution contains, the more relevant it tends to be. Furthermore, to prevent high relevance scores in case of duplicated facts in the solution, we perform an LLM-based duplication check. When evaluating the relevance of a single consequence, we propose two methods. First, we employ an LLM-as-a-judge strategy to score the relevance of each consequence on a scale of 1 to 10, based on the amount of information it carries. The underlying zero-shot prompt is given below. For brevity, we neglect the structured hybrid generation.

### **Prompt**

The following text contains a scenario and a legal consequence that was derived from the application of laws.

Your task is to evaluate the relevance of this consequence based on the following criterion: How much new information does the consequence contribute to the scenario?

Assign a value between 1 (least new information) and 10 (most new information) and print it at the end of your answer after 'Value:'.

**Payload**

```
### Contract
Cooperation Agreement ...
```

```
### Scenario
Max is currently...
```

```
### Consequence
Max has ...
```

Apart from this, the depth of the grounding tree for a given consequence is leveraged as a relevance score. The rationale behind this is that the higher the grounding tree of a fact, i.e. the longer the longest reasoning chain, the more difficult it is to derive. While the LLM-as-a-judge method directly scores the relevance of a fact based on its amount of information, this strategy assesses the non-evidence of a consequence. In other words, a high grounding tree depth indicates a non-evident consequence since more reasoning steps are involved.

In summary, during the evaluation of *Chain-of-Regulation* and its Chain-of-Thought baseline, we report the average values of both relevance scores. In addition, to directly verify the performance of our approaches in terms of the trade-off between hallucination mitigation and relevance, we report what we denote as the relevance-weighted factuality score. For a given solution, it reports the proportion of relevance scores that are, according to the recursive grounder, associated with grounded facts. Equation 5.1 specifies the relevance-weighted factuality score for a set of legal consequences, denoted by  $LC$ .  $R_{LLM}(c)$  is the LLM-based relevance score for consequence  $c$ , while  $F_{rec}(c)$  is the binary factuality for  $c$  according to the recursive grounder.

$$F(LC) = \frac{\sum_{c \in LC} R_{LLM}(c) * F_{rec}(c)}{\sum_{c \in LC} R_{LLM}(c)} \quad (5.1)$$

### 5.3 Evaluating Hallucination Mitigation

In this section, we describe the evaluation of *Chain-of-Regulation* in terms of hallucination mitigation, providing a quantifiable answer to the fundamental research question of this thesis: to what extent this approach can reduce hallucination. In particular, as our approach involves combining four different decomposition levels with four varying

intermediate regulation methods, including no regulation at all, we evaluate 16 different strategies to solve the multi-hop legal reasoning task. The evaluation is based on the recursive grounder. Additionally, we conduct shallow grounding, cross-comparisons, and both relevance assessments.

The evaluation set is based on the 54 non-disclosable legal contracts. For each of these contracts, we generate a concrete scenario and retrieve a set of relevant legal documents with our RAG system. More details on the task setup can be found in Section 4.1. In summary, our evaluation set comprises 54 samples, each consisting of a contract, a scenario, and a set of legal documents, ensuring that every approach is evaluated using the same samples. We apply every approach three times to the same data samples to allow cross-comparisons between solutions. As illustrated in Equation 5.2, this results in 2592 solutions to the multi-hop legal reasoning task. Due to computational restrictions, we restrict the size of the evaluation set to 54.

$$16 \text{ approaches} \times 3 \text{ repetitions} \times 54 \text{ samples} = 2592 \text{ runs} \quad (5.2)$$

It is important to note that we use none of the contract-based samples during the development of the *Chain-of-Regulation* approaches and their baseline to prevent data leakage. Specifically, using the evaluation dataset for both the development of the approaches and their evaluation would lead to biased results and undermine their generalization. Instead, we use synthetic data during development, with data samples consisting of a concrete scenario and a set of implication rules expressed in natural language. The data is generated using a one-shot example. The data structure is presented below.

```

1 {
2   "scenario": "FooBar Ltd. registers a new trademark...",
3   "implication_rules": "If a trademark is similar to ...",
4   "type": "local_rules",
5   "id": 0
6 }
```

The data used during development is generated using Gemini 2.5 Pro. In addition, we generate a small set of samples without a predefined set of implication rules. Instead, we retrieve rules using the RAG system. These examples are then associated with type `rag_rules_easy`.

## 5.4 Determining the Development of Hallucination

As introduced in Section 4.5, we examine the development and propagation of hallucination in the multi-hop reasoning setting from two perspectives: a qualitative perspective, which focuses on error analysis, and a quantitative study, which observes hallucination scores

ground-truth size	251	232	224	242	175	51
reasoning level	1	2	3	4	5	6

Table 5.2: Number of correct consequences for each reasoning level in the synthetic dataset used to observe the development of hallucination.

for varying reasoning levels. We collect results for both perspectives using a ground-truth-based evaluation, which requires the creation of another synthetic dataset. In this section, we describe the generation of this dataset and its use in capturing the development and propagation of hallucinations.

For the quantitative evaluation, the dataset must be structured to allow for a reasoning level-based evaluation of each consequence. To achieve this, we compose samples with a set of correct consequences, where each consequence is associated with its reasoning level. Hence, all consequences that are not contained in this set are considered to be hallucinated. This sample structure is also suitable for the qualitative analysis, where the factuality of each fact is encoded using colors in the reasoning tree visualization. This distinction between hallucinated and factual facts is essential to identify subsequent errors visualized in the reasoning trees. The following example illustrates the resulting sample structure.

```

1  {
2      "scenario": "Max is currently ...",
3      "correct_consequences": [
4          "Between Max and ...",
5          "...",
6      ],
7      "entailment_depths": {
8          "Between Max and ...": 1,
9          "...": "...",
10     },
11     "implication_rules": [
12         "<Implication>...</Implication>",
13         "...",
14     ]
15 }

```

The dataset is generated using Gemini 2.5 Pro with a prompt enhanced by a one-shot example. We aim for balanced consequences, i.e., they are equally distributed among the reasoning levels 1 to 6. While this is approximately the case for the first five reasoning levels, we do not achieve as many examples for reasoning level 6. Table 5.2 documents the exact distribution of consequences among the reasoning levels. In general, the dataset consists of 103 samples with a total of 1175 correct consequences as ground-truth.

# Results & Discussion

This chapter presents the results of this thesis, which enable us to address its underlying research questions. First, Section 6.1 gives detailed results of the recursive grounder evaluated as a binary classifier to detect hallucination. As such, this section answers Research Question 1.3. Then, in Section 6.2, we address the results of the actual approach, which provide a quantifiable answer to our fundamental research question: to what extent *Chain-of-Regulation* can reduce hallucination. It also gives a combined answer to Research Questions 1.1 and 1.2. In addition, we discuss the performance of our approaches in terms of efficiency. Section 6.3 contains the results of the analysis of hallucination development from both a qualitative and a quantitative aspect, which refer to Research Question 1.4. The underlying evaluation setup is explained in chapter 5.

## 6.1 Grounding

This section provides the results of the recursive grounder, which is the primary source of truth during the evaluation of *Chain-of-Regulation* in terms of hallucination mitigation. Hence, it answers Research Question 1.3, which asks to what extent we can detect hallucinations in our setting. We compare this approach to a baseline implementation, which verifies facts using a Chain-of-Thought strategy, the shallow grounder. In addition, we also report the efficiency in terms of token usage and their consistency, as both methods are non-deterministic. Finally, we discuss grounding trees, which are provided by the recursive grounder for each fact it verifies.

### 6.1.1 Hallucination Detection

The capability of the recursive grounder to detect hallucination in a setting that consists of a set of implication rules and a set of facts is evaluated using a ground-truth-based evaluation dataset. Specifically, we assess it as a binary classifier to decide whether a fact

is hallucinated or factual. For more details on the evaluation setup, refer to Section 5.1. Figure 6.1 illustrates the confusion matrix of the recursive grounder. We also evaluate the shallow grounder under this setting, which results in the confusion matrix shown in Figure 6.2. Furthermore, Table 6.1 summarizes important classification metrics. Note that factual facts are considered to be the positive class, whereas hallucinated facts represent the negative class.

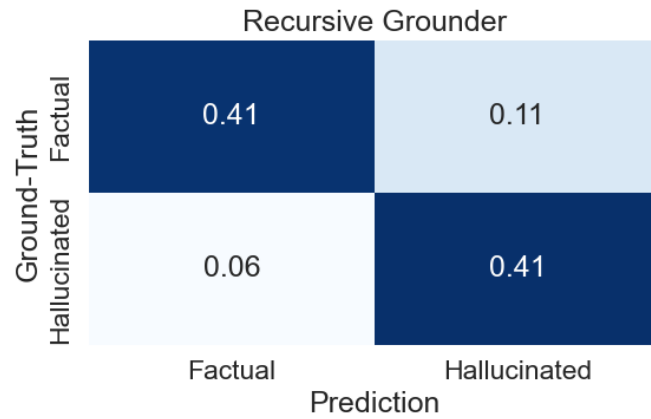


Figure 6.1: The confusion matrix of the recursive grounder evaluated as a binary classifier to verify the factuality of facts.  $n = 630$

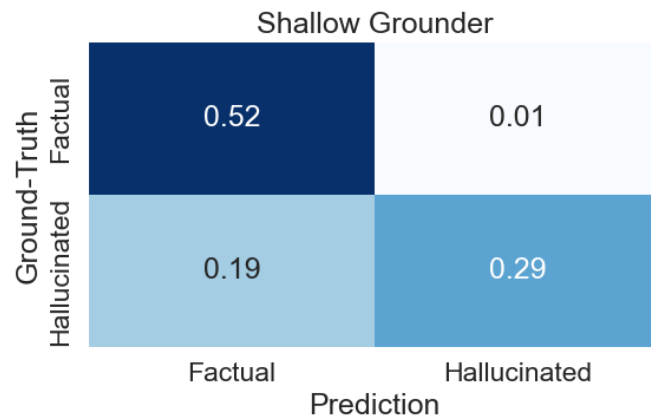


Figure 6.2: The confusion matrix of the shallow grounder evaluated as a binary classifier to verify the factuality of facts.  $n = 630$

From the results, we can see that the recursive grounder outperforms the shallow grounder in general. Intuitively, the recursive grounder is more balanced, i.e., it has no significant bias towards one of the classes, resulting in 82% overall accuracy. As such, 6% of all cases are false positives and 11% are false negatives. In contrast, the shallow grounder is too optimistic, resulting in 19% false positives and only 1% false negatives, where the

Metrics	Recursive Grounder	Shallow Grounder
Accuracy	<b>0.82</b>	0.8
Precision	<b>0.87</b>	0.73
Recall	0.78	<b>0.99</b>
Specificity	<b>0.87</b>	0.6
False Positive Rate	<b>0.13</b>	0.4
False Negative Rate	0.22	<b>0.01</b>

Table 6.1: Comparison between shallow grounder and recursive grounder in terms of classifying factuality based on our synthetic ground-truth.

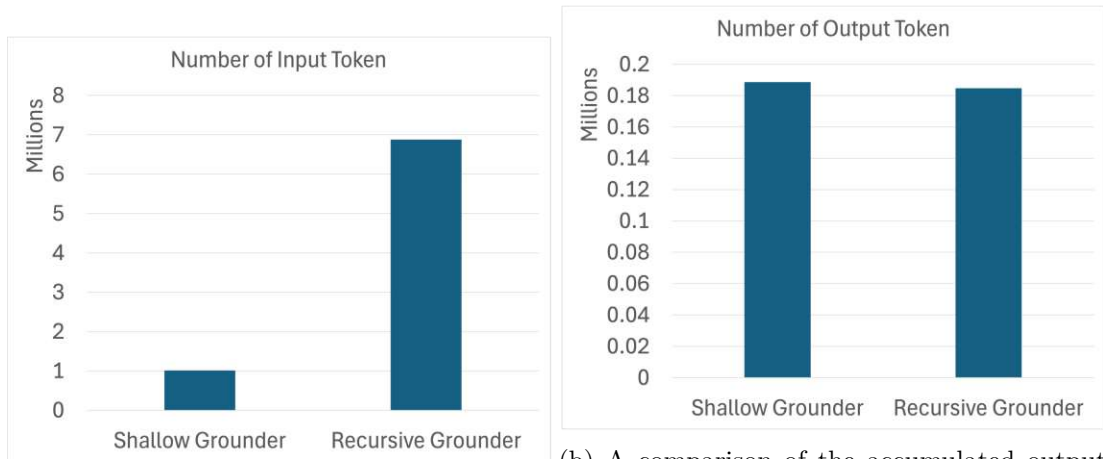
former refers to hallucinated facts that are incorrectly marked as factual, and the latter relates to correct facts that are incorrectly flagged as ungrounded.

Specifically, we emphasize the importance of the false positive rate when evaluating a hallucination detection method, as it indicates the proportion of hallucinated facts it fails to uncover. This is the most critical case as it gives hallucinated facts an additional sense of legitimacy. Also in this context, the shallow grounder, with a false positive rate of 0.4, is outperformed by the recursive grounder, which achieves 0.13. Likewise, the classifier’s precision indicates the proportion of truly factual facts among all facts marked as such. For the recursive grounder, this results in 87% precision in comparison to 73% for its shallow counterpart.

Except for recall and false positive rate, for which the shallow grounder achieves excellent values due to its bias towards the positive class, the recursive grounder outperforms the shallow benchmark in all other binary classification metrics. In the absence of superior alternative classifiers and a ground-truth, this justifies its use as a reference method during the evaluation of *Chain-of-Regulation*. Nevertheless, it is essential to note that the reported metrics are based on a synthetic ground-truth, as manual validation by legal experts is too costly. We conclude that the shallow grounder is too naive, which may be caused by its internal reasoning chain, on which it is based. For each fact, it reasons step-by-step, in a Chain-of-Thought manner, to conclude that the given statement is either factual or hallucinated. It seems that the shallow grounder "pieces something together" to wrongly conclude that hallucinated facts are factual.

### 6.1.2 Grounder Efficiency

In addition to the effectiveness of the recursive grounder, we also verify its practical applicability, especially in terms of token usage. Therefore, we report the input token usage and output token usage separately, which is reasonable due to their differing impact on inference time. This evaluation is solely based on a comparison between the two grounder variants and hence we report accumulated values for the ground-truth-based evaluation dataset. Figure 6.3 compares the token usage for both variants.



(a) A comparison of the accumulated input token usage during grounder evaluation.  $n = 630$   
 (b) A comparison of the accumulated output token usage during grounder evaluation.  $n = 630$

Figure 6.3: A comparison between token usages of the recursive grounder and the shallow grounder.

Token Usage	Recursive Grounder	Shallow Grounder
Input	10915.7	1602.8
Output	293.1	299.3

Table 6.2: Average token usage for the verification of a single fact in the evaluation data set.

In terms of input token usage, the recursive grounder is circa seven times more expensive than its shallow alternative, whereas both require approximately the same number of output tokens. While this indicates that the shallow grounder outperforms the recursive grounder in terms of efficiency, we argue that the output token usage is a lot more critical than the input token usage. This is because the inference time and, thus, the computational costs of an LLM are mostly dependent on the number of output tokens, as each token is generated sequentially in an autoregressive manner. Therefore, we argue that the recursive grounder is computationally feasible. The average costs to verify a single fact in the evaluation dataset are given in Table 6.2.

### 6.1.3 Grounder Consistency

The third type of evaluation we conduct for both grounders is to determine what we refer to as the consistency of the hallucination detection methods. Specifically, both grounders are LLM-driven, and since we use a non-zero temperature, the underlying decoding strategy is non-deterministic. The model parameters are specified in Table 4.2 and remain unchanged throughout the entire evaluation process. We perform the

grounding multiple times for a reduced set of facts to verify the consistency of both grounders.

Specifically, the consistency score of a grounder  $g$ , denoted by  $C(g)$ , is based on the function  $c_{10}(f, g)$ , which takes a single fact  $f$  and a grounder  $g$ . The grounder function is represented by a binary function, which returns either 0 or 1. The function  $c_{10}(f, g)$  takes the average of the grounder decision over ten repetitions, as specified in Equation 6.1. Based on this average value, we can now determine the grounder's consistency since an average of 0.5 indicates random results, whereas a score near 0 or 1 signals consistency. According to these considerations, Equation 6.2 computes the consistency value for a single fact  $f$  and then averages over all the facts in the evaluation set  $F$ . Thus, the closer the value is to 1, the better the consistency, whereas a score of 0.5 indicates random grounding. Figure 6.4 illustrates the consistency scores of both grounders.

$$c_{10}(f, g) = \frac{\sum_{i=1}^{10} g(f)}{10} \quad (6.1)$$

$$C(g) = \frac{\sum_{f \in F} \max(c_{10}(f, g), 1 - c_{10}(f, g))}{|F|} \quad (6.2)$$

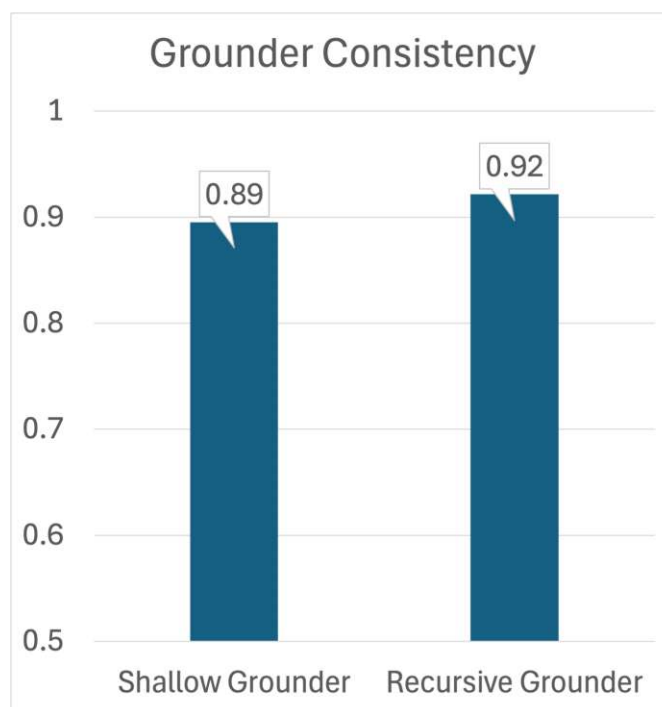


Figure 6.4: A comparison of consistency scores for both grounding components.  $n = 171$

Surprisingly, the recursive grounder with a score of 0.92 is slightly more consistent than the shallow grounder with a score of 0.89. Even though the recursive grounder involves

a set of interdependent LLM generations with many conditional recursive reasoning steps, which can be seen as the expansions of the grounding tree, this approach is more consistent than the single generation involved in the shallow grounder. This may be due to the flexibility of the internal reasoning process underlying the shallow grounding, which does not traverse implication rules and their concretized premises as systematically as the recursive approach. Apart from that, both grounding components achieve high consistency values.

### 6.1.4 Grounding Trees

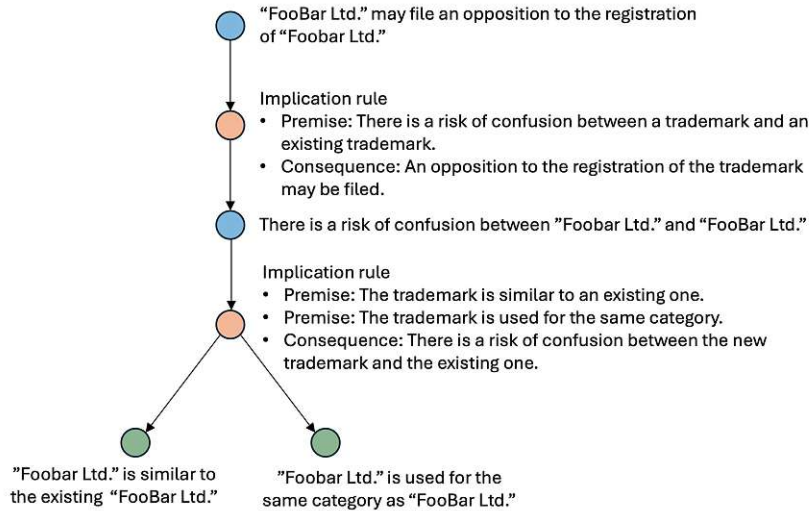
Another aspect of the recursive grounder, which does not directly contribute to answering the research questions of this thesis, is its capability to provide what we refer to as grounding trees. This is a visual representation illustrating the grounder’s traversal of implication rules to verify a fact until the underlying premises are directly grounded in the scenario or considered to be hallucinated. This allows a manual verification of the grounding process. In addition, even though it does not necessarily represent the actual reasoning employed by a multi-hop legal reasoner, as there may be multiple reasoning chains to derive a fact, it enables us to justify the applied reasoning for a given consequence. During the evaluation of *Chain-of-Regulation*, we compose grounding trees for each legal consequence. In the event of a grounding failure, the grounding tree indicates the incomplete grounding and identifies the specific premise where it occurred. Figure 6.5 illustrates grounding tree examples for both cases: a successful verification and the grounding process of a hallucinated fact. Both scenarios are fictional.

## 6.2 *Chain-of-Regulation*: Regulations $\times$ Decomposition Levels

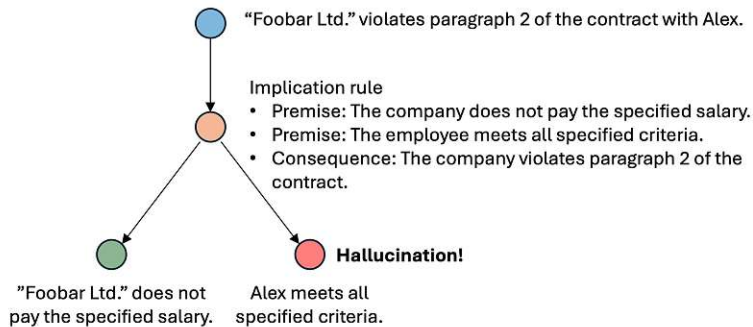
This section documents the results of *Chain-of-Regulation* in terms of hallucination mitigation, providing an answer to the underlying question of this thesis: to what extent can a combination of task decomposition and intermediate subtask regulation reduce hallucination in answers to the multi-hop legal reasoning task? Our answer is based on a comparison to a popular prompting-based hallucination reduction technique, namely, the Chain-of-Thought technique. Furthermore, to give detailed answers to Research Questions 1.1 and 1.2, we compare different decomposition levels, varying intermediate regulation approaches, and their combinations, resulting in 16 separate methods, including the CoT baseline. In addition, we also report the efficiency of each approach to verify its computational feasibility.

### 6.2.1 Hallucination Mitigation & Relevance

As addressed in Section 5.3, we evaluate each of the approaches over the set of non-disclosable legal contracts. Specifically, each of them is applied three times to the same data sample, all of which comprises a scenario and a legal contract. Then, the results,



(a) The grounding tree of a successful grounding process.



(b) The incomplete grounding tree associated with the grounding process of a hallucinated fact.

Figure 6.5: Grounding tree examples.

namely, the set of legal consequences, are verified by the recursive grounder, the shallow grounder, and a cross-comparison, where the recursive grounding is considered as the main baseline. We report the hallucination ratio as the most important metric, which specifies the proportion of hallucinated facts contained in a solution. Table 6.3 compares this metric between all our approaches according to the recursive grounder.

The Chain-of-Thought benchmark is the combination of no regulation and the monolithic decomposition. First, from the results, we can see that the lower the decomposition level, the better the results, except for the high-level decomposition. This level does not achieve significant improvements over the Chain-of-Thought benchmark, even if intermediate regulation is applied. Also, regulating the outputs of the monolithic approach in fact results in improvements, especially in the case of naive mitigation, which reduces the hallucination ratio by 20% compared to the CoT benchmark. In contrast, for the

<i>Chain-of-Regulation</i>	No Regulation	Task Specific	Naive Mitigation	Hallucination Specific
Monolithic	0.349	0.350	0.284	0.314
High-Level	0.391	0.347	0.392	0.366
Medium-Level	0.210	0.162	<b>0.134</b>	0.201
Low-Level	0.204	0.207	0.174	0.225

Table 6.3: Comparison of the hallucination proportion of every applied approach according to the recursive grounder.

two lower-level decomposition levels, the intermediate regulations achieve significant improvements. For instance, the naive mitigation also reduces the hallucination of the medium-level decomposition from 0.210 to 0.134, representing a reduction of almost 40%. Interestingly, this improvement is smaller at the low level, where the best reduction achieved by intermediate regulation is only about 15%. As for the comparison of intermediate regulation techniques, it is surprising that the task-specific regulation, which is the most sophisticated, is outperformed by low-effort regulations in many cases.

We can draw three important conclusions from these observations. The first one refers to the task decomposition, as the hallucination scores improve with increasing decomposition. Hence, we can conclude that the underlying tasks are complex for an LLM, and our decompositions effectively reduce this complexity. Also, we can conclude that the design of the decomposition, especially the interfaces between subtasks, is effective. Furthermore, the application of intermediate regulation yields better results at the low and medium levels than at the monolithic and high levels. Therefore, we conclude that the refinement of outputs is more effective when applied throughout the entire solution process rather than solely to the final outputs. The CoT baseline produces 35% hallucinated facts, whereas the best *Chain-of-Regulation* combination, namely the medium-level decomposition with a naive hallucination mitigation strategy, achieves a decrease in hallucination to approximately 13%. This allows us to answer the research question of this thesis: In comparison to the famous CoT baseline, *Chain-of-Regulation* achieves a reduction in the amount of hallucination by a factor of approximately 2.7.

Table 6.4 contains the results according to the shallow grounder. The results correlate with those of the recursive grounder to some extent, such that the best approach is again the use of naive mitigation on the medium decomposition level. It fails to identify any hallucination in the answers of this approach. This correlation allows us to conclude that the recursive grounder is not biased towards the *Chain-of-Regulation* approaches, as, according to the shallow grounder, the CoT benchmark hallucinates the most again. As already discussed in Section 6.1, the shallow grounder is too naive, and hence it reports overall lower proportions of hallucination.

Table 6.5 shows the results of the cross-comparison. The values are between 0 and 1. High scores indicate a high self-consistency of answers, which in turn implies less

<i>Chain-of-Regulation</i>	No Regulation	Task Specific	Naive Mitigation	Hallucination Specific
Monolithic	0.009	0.007	0.001	0.003
High-Level	0.007	0.005	0.002	0.001
Medium-Level	0.003	0.001	<b>0.000</b>	0.003
Low-Level	0.005	0.005	0.004	0.005

Table 6.4: Comparison of the hallucination proportion for every applied approach according to the shallow grounder.

<i>Chain-of-Regulation</i>	No Regulation	Task Specific	Naive Mitigation	Hallucination Specific
Monolithic	0.74	0.73	0.57	0.74
High-Level	0.73	0.68	0.75	0.76
Medium-Level	0.79	0.77	<b>0.80</b>	0.79
Low-Level	0.72	0.72	0.73	0.74

Table 6.5: Comparison of the average self-consistency scores for every applied approach according to the embedding-based cross-comparisons. High values indicate better self-consistency.

hallucination. Again, the results correlate to some extent with the results of the recursive grounder. Thus, the naive mitigation applied to the medium-level decomposition achieves a self-consistency of 0.8, which is the highest value. In general, this decomposition level achieves higher self-consistency than the others. Intermediate regulation does not contribute to more self-consistency. This may be because regulation results in longer sequences of subtasks, which additionally manipulate intermediate outputs, causing even less deterministic results.

As argued in Section 5.2, there may be a trade-off between result relevance and hallucination mitigation. In summary, a model may prioritize factuality over relevance. To observe whether this is the case, we report the average tree depth in Table 6.6 and the average relevance scores, judged by an LLM, in Table 6.7. While the relevance scoring takes facts that are hallucinated according to the recursive grounder into account, we define the tree depth of a hallucinated fact to be zero.

According to the average grounding tree depth, there is indeed a trade-off between relevance and actuality, as the best combination in terms of hallucination mitigation, namely, naive mitigation applied on the medium-level decomposition, is among the worst when it comes to the average tree depth. In addition, the converse is also true, as the CoT benchmark provides the most relevant consequences in this aspect. This is also in part underpinned by the LLM-driven relevance evaluation. Although the values show relatively low variance, we can observe slightly lower relevance values for the lower levels

<i>Chain-of-Regulation</i>	No Regulation	Task Specific	Naive Mitigation	Hallucination Specific
Monolithic	<b>2.45</b>	2.29	2.16	2.16
High-Level	2.14	1.99	2.03	2.22
Medium-Level	1.78	1.73	1.73	1.72
Low-Level	2.16	1.83	1.88	1.92

Table 6.6: Average grounding tree depth of factual consequences for each approach.

<i>Chain-of-Regulation</i>	No Regulation	Task Specific	Naive Mitigation	Hallucination Specific
Monolithic	4.54	4.6	3.61	4.51
High-Level	5.11	<b>5.24</b>	4.97	4.97
Medium-Level	4.52	4.58	4.29	4.14
Low-Level	4.14	4.13	4.13	4.26

Table 6.7: Average relevance scores of the consequences for each approach according to an LLM-as-a-judge.

of decomposition in comparison to the Chain-of-Thought benchmark.

Table 6.8 gives the relevance-weighted factuality, which is specified in Equation 5.1 and allows for evaluating the performance of *Chain-of-Regulation* in terms of the trade-off using a single score. Intuitively, it reports the proportion of factual relevance for a given answer. Again, the combination of medium-level decomposition with naive regulation achieves the best performance, with 83% of the generated relevance being factual, whereas the CoT benchmark scores solely 64%. As this score strongly correlates with the hallucination proportion by design, the deviation from this score is particularly interesting. Hence, 17% of the accumulated relevance of the best combination is considered to be hallucinated, in contrast to 13% of the consequences. This slight deviation suggests that factual consequences are less significant than hallucinated ones. In comparison, for CoT, we report 36% with relevance weighting and 35% without. Hence, our best approach prioritizes factuality slightly more than CoT, neglecting relevance.

### 6.2.2 Efficiency

In addition to its effectiveness in terms of hallucination mitigation, we also conduct an efficiency evaluation of our approaches, as in practice, there is a trade-off between computational resources and the quality of the output. Specifically, we report the input token usage and output token usage separately for each method. Splitting the token usage into input and output is reasonable due to their differing impact on inference time. Again, we use the zero-shot Chain-of-Thought approach as a benchmark. Table 6.9 reports the input token usage for each combination, and Table 6.10 reports the output

<i>Chain-of-Regulation</i>	No Regulation	Task Specific	Naive Mitigation	Hallucination Specific
Monolithic	0.64	0.65	0.53	0.69
High-Level	0.6	0.66	0.6	0.63
Medium-Level	0.75	0.8	<b>0.83</b>	0.76
Low-Level	0.75	0.74	0.78	0.71

Table 6.8: Average relevance-weighted factuality scores of the consequences for each approach.

<i>Chain-of-Regulation</i>	No Regulation	Task Specific	Naive Mitigation	Hallucination Specific
Monolithic	<b>12</b>	26	25	24
High-Level	82	76	105	111
Medium-Level	288	601	415	416
Low-Level	247	339	329	329

Table 6.9: Average input token usage for each approach. Values are given in  $10^3$ .

<i>Chain-of-Regulation</i>	No Regulation	Task Specific	Naive Mitigation	Hallucination Specific
Monolithic	<b>11</b>	123	25	14
High-Level	21	26	27	26
Medium-Level	31	156	55	59
Low-Level	171	147	220	123

Table 6.10: Average output token usage for each approach. Values are given in  $10^2$ .

token usage.

As can be seen from the results, *Chain-of-Regulation* is very expensive in terms of input token usage, which is particularly true for the two lower-level decompositions. This is because the majority of subtasks are provided with the entire task context, which includes the contract and the scenario. Interestingly, the low-level approach uses fewer input tokens than the medium-level approach, despite involving more subtasks. This may be because the lowest level decomposition processes premises one by one and terminates the application of an implication rule if a premise is not satisfied. In addition, the low-level approach simplifies the task context by reducing it to relevant facts.

Additionally, we can conclude that the application of both the naive and hallucination-specific regulations is relatively cost-effective, as in all cases they consume less than double the tokens used without regulation. In contrast, the task-specific regulation may be very computationally expensive, such as in the case of the medium-level decomposition.

However, they can also reduce the token consumption, such as for the high-level approach, which seems counterintuitive at first. This is because it improves the application of implication rules, such that, e.g., it adds additional consequences and hence achieves more legal consequences in a reduced number of subtask executions.

In comparison to the Chain-of-Thought benchmark, which is the most cost-effective approach, our most effective combination in terms of hallucination mitigation consumes approximately 35 times more input tokens on average. However, we do not consider this to be critical as the impact of the number of input tokens on the inference time is comparatively small.

In contrast, the number of output tokens is highly relevant to both computational costs and inference time. Hence, an increase in token usage for *Chain-of-Regulation* compared to Chain-of-Thought is considered critical. In general, we can see that the lower the decomposition level, the higher the output token usage, which is evident. Also, there are again two cases where the application of intermediate regulation reduces the number of generated tokens. Again, this can be explained by an improved multi-hop legal reasoning achieved by the regulation. Apart from these exceptions, intermediate regulation increases the token usage. Specifically, for the naive mitigation and the hallucination-specific regulation, this increase is bounded by a factor of 2.3, whereas for the task-specific regulation, token usage may exceed that of the unregulated approach by a factor of 11.2 in the worst case.

When it comes to the comparison of our best approach with the CoT benchmark, we observe an output token increase by a factor of 5. Therefore, in contrast to Chain-of-Thought, *Chain-of-Regulation* mitigates hallucinations by a factor of 2.7 for five times its output token costs. However, this increase affects relatively short outputs, as for our best approach, the average output length is 5.5k tokens for inputs that are on average 12k tokens long. In summary, we argue that *Chain-of-Regulation* is a competitive method to mitigate hallucination.

### 6.3 Development of Hallucination

The third part of the results provided by this thesis focuses on the development of hallucination in the multi-hop legal reasoning setting. Specifically, we distinguish between a quantitative aspect of this development, on the one hand, which we denote as hallucination development. In summary, we report the proportion of hallucinated facts over the reasoning level. The reasoning level of a fact is defined by the length of its longest reasoning chain. On the other hand, we approach this issue from a qualitative perspective, which investigates how hallucinations propagate. In essence, we manually identify errors that result in hallucinated facts using the reasoning tree visualization. It allows for the illustration of the reasoning process of the low-level decomposition level without intermediate regulation. These observations are restricted to this approach and use a ground-truth for each reasoning level. A detailed documentation of the underlying methodology can be found in Section 4.5.

### 6.3.1 Quantity: Hallucination Development

To observe how the proportion of hallucinated facts develops over the reasoning level, we use a synthetic dataset with a maximum reasoning level of 6. The dataset defines a set of correct consequences for each level such that, in case a fact of this level is not part of the set, it is considered to be hallucinated. Figure 6.6 shows the results.

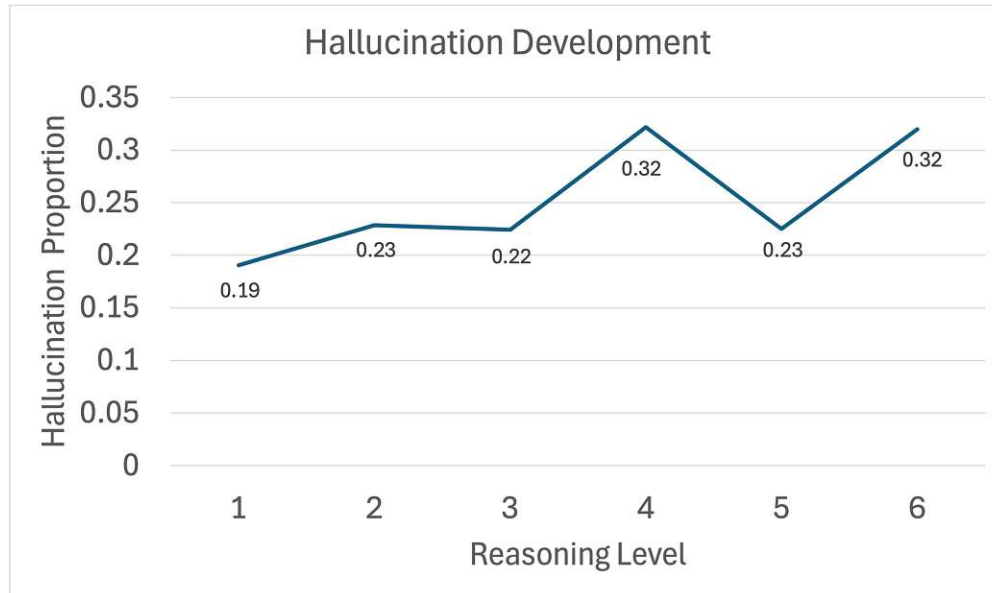


Figure 6.6: The development of the proportion of hallucinated facts over the reasoning level.

As expected, the proportion of hallucinations increases with higher reasoning levels. Starting from an initial score of ca. 19% hallucination, which is comparable to the result in Table 6.3 for this approach, it increases to 32% on the highest reasoning level. This is reasonable, as due to the interdependency of implication rules, caused by the multi-hop reasoning property, errors have an impact on subsequent reasoning levels. However, it seems that the LLM achieves restoring mistakes to some extent, as the hallucination development is not continuous and decreases for level 5. Hence, it appears that the context of the problem allows the LLM to sanitize, e.g., wrongly concretized premises, such that the derived consequence is valid again. Apart from this fluctuation, we can conclude that the consequences of high reasoning levels are more complex to derive and, hence, on average, less factual.

### 6.3.2 Quality: Hallucination Propagation

If we solely analyze the quantity of hallucination observed at a specific reasoning level, we fail to identify its causes. As a result, in addition to the previous findings, we also evaluate what we refer to as hallucination propagation. In summary, we visually represent the LLM's reasoning process to manually identify typical errors made by the model

and how they contribute to subsequent hallucinations. The underlying concept is the reasoning tree, which illustrates all facts and implication rules used by the approach to derive the legal consequences. Figure 6.7 gives an example. In general, they are related to grounding trees. However, there is a crucial difference, namely, the reasoning tree represents the actual reasoning used by the underlying approach. Factual consequences are highlighted in green, while red implies hallucination. In addition, the reasoning tree begins with the set of basic facts that comprise the initial scenario, indicated in blue. The example illustrates the confusion of concrete entities, resulting in a hallucinated consequence.

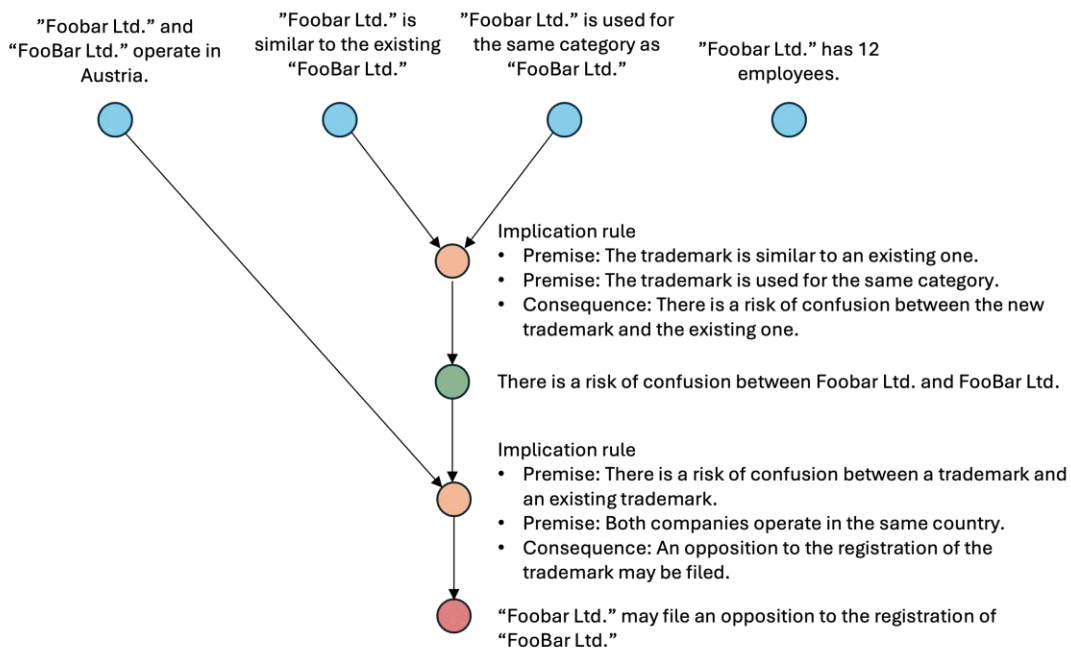


Figure 6.7: An example reasoning tree as used to identify typical errors made by the approach. It contains both a correct and a wrong implication rule application.

Specifically, in the multi-hop reasoning setting, we identify a set of specific errors resulting in hallucination, which are described in Table 6.11. The used approach is based on the low-level decomposition, with the exception that it only decides for each premise if it is relevant given an implication rule, rather than processing it into concretizations and assumptions. As this table is restricted to the most frequent errors we identified in the reasoning trees, it is not complete. Although this analysis is not necessary to answer the research question of this thesis, its results motivate task-specific regulation techniques. For instance, the implication rule extraction regulation verifies for each implication rule if its consequence is equal to one of its premises to avoid circular reasoning.

Error	Description
Wrong implication rule extraction	In fact, this refers to a set of different errors resulting in wrong implication rules. Note that the reasoning tree does not allow for the identification of all of these errors, as it does not contain the natural text describing the rule. A frequent pattern is that the consequence is incorrectly considered an additional premise, resulting in circular reasoning. Also, the logical relation between premises seems to be difficult for LLMs to identify, so that they, e.g., confuse logical conjunctions and disjunctions with each other.
Application of unsatisfied implication rule	A subtask takes a local set of context facts into account to decide whether a given implication rule can be applied. If this is not the case, the LLM is prompted to specify that the rule application is not possible. If the model still applies the rule, it derives a wrong consequence, which is considered to be hallucinated. For the regular low-level decomposition approach, this error can only occur during the premise processing, where the model verifies for each premise whether it is satisfied.
Wrong consequence concretization	In case all premises are met, i.e., they are contained in the scenario in a concretized manner, the model derives the new consequence based on these concretizations. It is required to take the relations of the abstract and concrete entities into account, as illustrated in Figure 4.6. This complexity is another frequent source of errors. For instance, if a premise is concretized in the scenario more than once, e.g., "FooBar Ltd. is a company," and "Foobar Inc. is a company.", the model has difficulty identifying the correct entity. Figure 6.7 illustrates this type of error.

Table 6.11: A summary of important errors that contribute to hallucination according to the reasoning tree analysis.



Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar  
The approved original version of this thesis is available in print at TU Wien Bibliothek.

# Conclusion & Future Work

This chapter presents the final conclusions of this thesis, which we approach by first summarizing the most important findings and then concluding how they answer the underlying questions of this work. In addition, we go beyond the scope of this work to discuss future work motivated by *Chain-of-Regulation*, which we categorize into application-focused and methodology-based aspects.

## 7.1 Conclusion

The fundamental goal of this thesis is to develop and evaluate a hallucination mitigation approach based on the combination of task decomposition and intermediate regulation, determining to what extent it can reduce hallucinations. Both the application and the evaluation of this method are restricted to what we introduce as the multi-hop legal reasoning task. The main baseline of this work is a zero-shot Chain-of-Thought strategy. We approach this problem by combining different levels of decomposition of the reasoning task with three self-refinement-based strategies to improve intermediate subtask results, resulting in 16 approaches. The evaluation is reference-free, utilizes a notion of hallucination that is based on factuality, and is conducted using a set of 54 legal contracts.

From the results of this setup, we can draw the fundamental conclusion that the combination of task decomposition and intermediate regulation with the purpose of mitigating hallucination in the final task result outperforms zero-shot Chain-of-Thought prompting in the multi-hop legal reasoning task by a factor of 2.7. Specifically, the baseline hallucination proportion of 35% can be reduced to 15% in the best case, namely the medium-level decomposition level with naive mitigation regulation.

Furthermore, we can conclude that the task decomposition itself is an effective method for mitigating hallucinations, as it achieves a reduction in hallucinations to 20.4% even

without intermediate regulation in the case of low-level tasks. In terms of the intermediate regulation, it is surprising that the approach that we consider the simplest outperforms the task-specific regulation, which involves the most effort. Several factors might explain this, including the discrepancy between subtask result optimization and hallucination mitigation in the final output as discussed in Section 4.3. Moreover, the complexity of task-specific regulation programs, i.e., the number of involved subtasks, can cause responses to be repeatedly altered, so that the content of the generated facts gradually drifts away from the actual task context, which results in hallucination.

The efficiency of *Chain-of-Regulation* significantly depends on the exact combination in use. In particular, the output token usage varies between a factor of 1.3 and 20 relative to that of the Chain-of-Thought benchmark. The most effective combination generates five times more output tokens than the baseline. In summary, *Chain-of-Regulation* is effective in reducing hallucination for the multi-hop legal reasoning task and is still sufficiently efficient.

Apart from the main results, we conclude that our recursive grounder is an efficient and effective method to identify hallucination in the multi-hop legal reasoning setting. Specifically, we evaluate it as a binary classifier to determine the extent to which it can detect hallucinations, which is its motivating question. It achieves an accuracy of 82% and outperforms the baseline implementation, a shallow grounding prompt, in both classification accuracy and output token usage. In the context of hallucination development, the results support our assumption that the proportion of hallucinations increases with the reasoning level. In addition, reasoning trees are a beneficial representation for identifying typical errors made during the reasoning task. Moreover, they allow for a better explainability of the generated results.

### 7.2 Future Work

There are several aspects that go beyond the scope of this thesis, which we address in this section to motivate future work. On the one hand, we identify what we refer to as application-based future work, i.e., required adaptations to turn our problem solution into a real-world application in the legal domain that can be used to review legal contracts. First, our first-come, first-served rule selection mechanism needs to be replaced with a more sophisticated approach that involves actual legal principles, e.g., *lex specialis derogat legi generali* [Gro20], which prioritizes specific law over general law. As such, it is one of many collision rules used by legal experts. In addition, references to other legal texts are not resolved by our implication rule extraction, which may be an issue for a reasonable application of law. To overcome this limitation, a knowledge graph representation appears to be a suitable approach. Furthermore, for an accurate application of law, the source of legal texts needs to be expanded to the hierarchical structure of the Austrian legal system, as introduced in Section 2.2. Finally, making the application multi-lingual is left for future work.

On the other hand, we identify methodology-based future work, which can be seen as an

extension of the methods employed in this thesis. First, regarding our setting formalization, which models the multi-hop reasoning task context as a set of implication rules and a set of atomic facts, we suggest generalizing it to other highly formalizable domains, such as physics. This would enable the construction of reasoners that consider knowledge from multiple domains. Furthermore, the formalization may also be extended to include logical relationships, which are common in NLI. Specifically, besides implication rules, future work may define additional concepts involving taxonomy relations and sequential reasoning as used in [DJT<sup>+</sup>22].

In addition, the recursive grounding can be improved by utilizing an RAG system when searching for applicable implication rules. As such, it could dynamically retrieve natural text containing implication rules before extracting them into our structured representation and using them during the grounding process. Finally, in terms of task decomposition, we suggest defining a generic set of subtasks that can be split into arbitrary tasks. In particular, the underlying idea is to devise a set of fundamental tasks, such as fact atomization, which enables the implementation of any task if correctly arranged. For each of these generic subtasks, you can then define a dedicated intermediate regulation.



Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar  
The approved original version of this thesis is available in print at TU Wien Bibliothek.

# Overview of Generative AI Tools Used

Generative AI was used to enhance the spelling and grammar of this thesis. Furthermore, we used it to generate LaTeX formatting templates for figures, algorithms, and tables.

Besides their scientific use, we utilized LLMs to generate simple utility functions, such as eliminating duplicates in a list, and visualization code during the development of the actual thesis. The affected code sections are marked accordingly. The following tools were in use:

- ChatGPT - Accessed from April 1, 2025 to September 30, 2025
- Grammarly - Accessed from August 1, 2025 to September 30, 2025



# List of Figures

2.1	An illustration of how tokens are processed in an encoder-decoder architecture and a decoder-only model [FLY <sup>+</sup> 23]. . . . .	6
2.2	A taxonomy describing important causes of hallucination [Cos25, HYM <sup>+</sup> 25, JLF <sup>+</sup> 23]. . . . .	10
2.3	The hallucination rate of LLMs over the prominence of cases addressed by the tasks. The prominence of a case is measured by PageRank percentile. The decline of hallucination rates with increasing prominence indicates a knowledge boundary [DMSH24]. . . . .	11
2.4	A high-level illustration of the legal reasoning task. . . . .	16
2.5	An example of a fact concretization in the context of a scenario s. . . . .	18
2.6	The application of an implication rule can be performed explicitly using its structured representation or implicitly if the rule is provided in natural language. . . . .	19
3.1	Taxonomy of hallucination mitigation strategies according to [TZJ <sup>+</sup> 24]. <i>Chain-of-Regulation</i> can be seen as Self-Refinement through Feedback and Reasoning method. . . . .	24
4.1	A conceptual overview of the core idea of <i>Chain-of-Regulation</i> , namely a combination of task decomposition with intermediate subtask result regulation. . . . .	31
4.2	A simplified scenario that illustrates the interactions of the RAG components during indexing and retrieval. . . . .	36
4.3	An overview of the task setup that involves the generation of a scenario, search terms, and queries to retrieve relevant legal documents. This setup is required to provide the context of <i>Chain-of-Regulation</i> . . . . .	37
4.4	The combination of intermediate regulation and different decomposition levels results in 16 methods, which comprise our <i>Chain-of-Regulation</i> approaches and their baselines. . . . .	59
4.5	This control flow illustrates how task-specific regulation utilizes structured output to reduce the complexity of regulation tasks. In this case, this allows to separately refine structured implication rules extracted from natural language. . . . .	61
4.6	The concretization of the premises of an implication rule involves three conceptual steps, which are illustrated in this example using two concrete entities. . . . .	69
		103

6.1	The confusion matrix of the recursive grounder evaluated as a binary classifier to verify the factuality of facts. $n = 630$ . . . . .	82
6.2	The confusion matrix of the shallow grounder evaluated as a binary classifier to verify the factuality of facts. $n = 630$ . . . . .	82
6.3	A comparison between token usages of the recursive grounder and the shallow grounder. . . . .	84
6.4	A comparison of consistency scores for both grounding components. $n = 171$	85
6.5	Grounding tree examples. . . . .	87
6.6	The development of the proportion of hallucinated facts over the reasoning level. . . . .	93
6.7	An example reasoning tree as used to identify typical errors made by the approach. It contains both a correct and a wrong implication rule application.	94

# List of Tables

4.1	Model comparison based on environmental constraints and required properties.	32
4.2	Important parameters for Gemma 3 27b specified by Ollama. . . . .	33
4.3	The set of predefined search terms used to index potentially relevant legal documents fetched from the RIS API. . . . .	35
4.4	The general structure of the used prompts is defined by a system message, the actual prompt, and a user message which contains the required context as payload. . . . .	39
4.5	Important concepts and a summary of their descriptions, as used in our prompts . . . . .	40
4.6	This table summarizes the subtask-specific regulation methods used to improve the results of the most important subtasks. . . . .	62
5.1	Number of correct consequences for each reasoning level in the grounder evaluation dataset. . . . .	76
5.2	Number of correct consequences for each reasoning level in the synthetic dataset used to observe the development of hallucination. . . . .	80
6.1	Comparison between shallow grounder and recursive grounder in terms of classifying factuality based on our synthetic ground-truth. . . . .	83
6.2	Average token usage for the verification of a single fact in the evaluation data set. . . . .	84
6.3	Comparison of the hallucination proportion of every applied approach according to the recursive grounder. . . . .	88
6.4	Comparison of the hallucination proportion for every applied approach according to the shallow grounder. . . . .	89
6.5	Comparison of the average self-consistency scores for every applied approach according to the embedding-based cross-comparisons. High values indicate better self-consistency. . . . .	89
6.6	Average grounding tree depth of factual consequences for each approach. .	90
6.7	Average relevance scores of the consequences for each approach according to an LLM-as-a-judge. . . . .	90
6.8	Average relevance-weighted factuality scores of the consequences for each approach. . . . .	91
6.9	Average input token usage for each approach. Values are given in $10^3$ . . .	91
		105

6.10 Average output token usage for each approach. Values are given in $10^2$ . . . . .	91
6.11 A summary of important errors that contribute to hallucination according to the reasoning tree analysis. . . . .	95

# List of Algorithms

2.1	Idealized Recursive Grounding <i>idealGround</i> . . . . .	21
4.1	Theoretical Multi-Hop Legal Reasoner . . . . .	44
4.2	High-Level Decomposition Approach . . . . .	47
4.3	Medium-Level Decomposition Approach . . . . .	50
4.4	Low-Level Decomposition Approach . . . . .	55
4.5	Recursive Grounder <i>recGround</i> . . . . .	67
4.6	Embedding-Based Cross Comparison . . . . .	72



Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar  
The approved original version of this thesis is available in print at TU Wien Bibliothek.

# Bibliography

- [BCB16] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural Machine Translation by Jointly Learning to Align and Translate, 2016.
- [BFGS19] Luigi Bellomarini, Daniele Fakhoury, Georg Gottlob, and Emanuel Sallinger. Knowledge Graphs and Enterprise AI: The Promise of an Enabling Technology. In *2019 IEEE 35th International Conference on Data Engineering (ICDE)*, pages 26–37, 2019.
- [BGW<sup>+</sup>24] Ryan C. Barron, Ves Grantcharov, Selma Wanna, Maksim E. Eren, Manish Bhattarai, Nicholas Solovyev, George Tompkins, Charles Nicholas, Kim Ø. Rasmussen, Cynthia Matuszek, and Boian S. Alexandrov. Domain-Specific Retrieval-Augmented Generation Using Vector Stores, Knowledge Graphs, and Tensor Factorization, 2024.
- [BQH<sup>+</sup>23] Farima Fatahi Bayat, Kun Qian, Benjamin Han, Yisi Sang, Anton Belyi, Samira Khorshidi, Fei Wu, Ihab F. Ilyas, and Yunyao Li. FLEEK: Factual Error Detection and Correction with Evidence Retrieved from External Knowledge, 2023.
- [CCZ<sup>+</sup>25] Yongming Chen, Miner Chen, Ye Zhu, Juan Pei, Siyu Chen, Yu Zhou, Yi Wang, Yifan Zhou, Hao Li, and Songan Zhang. Leverage Knowledge Graph and Large Language Model for Law Article Recommendation: A Case Study of Chinese Criminal Law, 2025.
- [Col24] Andrea Colombo. Leveraging Knowledge Graphs and LLMs to Support and Monitor Legislative Systems. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management, CIKM '24*, page 5443–5446. ACM, October 2024.
- [Cos25] Manuel Cossio. A comprehensive taxonomy of hallucinations in Large Language Models, 2025.
- [CRLB18] Oana-Maria Camburu, Tim Rocktäschel, Thomas Lukasiewicz, and Phil Blunsom. e-SNLI: Natural Language Inference with Natural Language Explanations, 2018.

- [CXZ<sup>+</sup>24] Jianlv Chen, Shitao Xiao, Peitian Zhang, Kun Luo, Defu Lian, and Zheng Liu. BGE M3-Embedding: Multi-Lingual, Multi-Functionality, Multi-Granularity Text Embeddings through Self-Knowledge Distillation, 2024.
- [DA25] DeepSeek-AI. DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning, 2025.
- [DJT<sup>+</sup>22] Bhavana Dalvi, Peter Jansen, Oyvind Tafjord, Zhengnan Xie, Hannah Smith, Leighanna Pipatanangkura, and Peter Clark. Explaining Answers with Entailment Trees, 2022.
- [DKX<sup>+</sup>23] Shehzaad Dhuliawala, Mojtaba Komeili, Jing Xu, Roberta Raileanu, Xian Li, Asli Celikyilmaz, and Jason Weston. Chain-of-Verification Reduces Hallucination in Large Language Models, 2023.
- [DMSH24] Matthew Dahl, Varun Magesh, Mirac Suzgun, and Daniel E. Ho. Large Legal Fictions: Profiling Legal Hallucinations in Large Language Models, 2024.
- [ea20] Tom B. Brown et al. Language Models are Few-Shot Learners, 2020.
- [ea24a] Aaron Grattafiori et al. The Llama 3 Herd of Models, 2024.
- [ea24b] Zhen Qin et al. Large Language Models are Effective Text Rankers with Pairwise Ranking Prompting, 2024.
- [ea25] Comanici et al. Gemini 2.5: Pushing the Frontier with Advanced Reasoning, Multimodality, Long Context, and Next Generation Agentic Capabilities, 2025.
- [Fil21] Erwin Filtz. *Knowledge Graphs for Analyzing and Searching Legal Data*. PhD thesis, Vienna University of Economics and Business (WU Vienna), Vienna, Austria, 2021.
- [FLY<sup>+</sup>23] Zihao Fu, Wai Lam, Qian Yu, Anthony Man-Cho So, Shengding Hu, Zhiyuan Liu, and Nigel Collier. Decoder-Only or Encoder-Decoder? Interpreting Language Model as a Regularized Encoder-Decoder, 2023.
- [FS23] Robert Friel and Atindriyo Sanyal. Chainpoll: A High Efficacy Method for LLM Hallucination Detection, 2023.
- [Gro20] Prof. Dr. P. Groschler. Lex specialis. Latein für Juristen, Folge 7, 2020. PDF available at: [https://groeschler.jura.uni-mainz.de/files/2020/06/LfJ-07-Lex\\_specialis.pdf](https://groeschler.jura.uni-mainz.de/files/2020/06/LfJ-07-Lex_specialis.pdf).
- [Gum25] Esmail Gumaan. Theoretical Foundations and Mitigation of Hallucination in Large Language Models, 2025.

- [HBB<sup>+</sup>21] Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring Massive Multitask Language Understanding, 2021.
- [HCF25] Justin Ho, Alexandra Colby, and William Fisher. Incorporating Legal Structure in Retrieval-Augmented Generation: A Case Study on Copyright Fair Use, 2025.
- [HLP<sup>+</sup>25] Geonho Hwang, Wonyeol Lee, Yeachan Park, Sejun Park, and Feras Saad. *Floating-Point Neural Networks are Provably Robust Universal Approximators*, page 301–326. Springer Nature Switzerland, 2025.
- [HSW<sup>+</sup>21] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-Rank Adaptation of Large Language Models, 2021.
- [HTZ<sup>+</sup>23] Quzhe Huang, Mingxu Tao, Chen Zhang, Zhenwei An, Cong Jiang, Zhibin Chen, Zirui Wu, and Yansong Feng. Lawyer LLaMA Technical Report, 2023.
- [HWS<sup>+</sup>25] Haoyu Han, Yu Wang, Harry Shomer, Kai Guo, Jiayuan Ding, Yongjia Lei, Mahantesh Halappanavar, Ryan A. Rossi, Subhabrata Mukherjee, Xianfeng Tang, Qi He, Zhigang Hua, Bo Long, Tong Zhao, Neil Shah, Amin Javari, Yinglong Xia, and Jiliang Tang. Retrieval-Augmented Generation with Graphs (GraphRAG), 2025.
- [HYM<sup>+</sup>25] Lei Huang, Weijiang Yu, Weitao Ma, Weihong Zhong, Zhangyin Feng, Haotian Wang, Qianglong Chen, Weihua Peng, Xiaocheng Feng, Bing Qin, and Ting Liu. A Survey on Hallucination in Large Language Models: Principles, Taxonomy, Challenges, and Open Questions. *ACM Transactions on Information Systems*, 43(2):1–55, January 2025.
- [IK20] Eleni Ilkou and Maria Koutraki. Symbolic Vs Sub-symbolic AI Methods: Friends or Enemies? 11 2020.
- [JLF<sup>+</sup>23] Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Ye Jin Bang, Andrea Madotto, and Pascale Fung. Survey of Hallucination in Natural Language Generation. *ACM Comput. Surv.*, 55(12), March 2023.
- [KKK24] Amna Khalid, Ayma Khalid, and Umar Khalid. The Role of Language Models in Modern Healthcare: A Comprehensive Review, 2024.
- [KR18] Taku Kudo and John Richardson. SentencePiece: A Simple and Language Independent Subword Tokenizer and Detokenizer for Neural Text Processing, 2018.

- [KSR<sup>+</sup>25] Muhammad Rafsan Kabir, Rafeed Mohammad Sultan, Fuad Rahman, Mohammad Ruhul Amin, Sifat Momen, Nabeel Mohammed, and Shafin Rahman. LegalRAG: A Hybrid RAG System for Multilingual Legal Information Retrieval, 2025.
- [KTF<sup>+</sup>23] Tushar Khot, Harsh Trivedi, Matthew Finlayson, Yao Fu, Kyle Richardson, Peter Clark, and Ashish Sabharwal. Decomposed Prompting: A Modular Approach for Solving Complex Tasks, 2023.
- [KWBH24] Timo Kaufmann, Paul Weng, Viktor Bengs, and Eyke Hüllermeier. A Survey of Reinforcement Learning from Human Feedback, 2024.
- [LGL<sup>+</sup>24] Qing Li, Jiahui Geng, Chenyang Lyu, Derui Zhu, Maxim Panov, and Fakhri Karray. Reference-free Hallucination Detection for Large Vision-Language Models, 2024.
- [LLH<sup>+</sup>23] Deren Lei, Yaxi Li, Mengya Hu, Mingyu Wang, Vincent Yun, Emily Ching, and Eslam Kamal. Chain of Natural Language Inference for Reducing Large Language Model Ungrounded Hallucinations, 2023.
- [LPC<sup>+</sup>21] Shayne Longpre, Kartik Perisetla, Anthony Chen, Nikhil Ramesh, Chris DuBois, and Sameer Singh. Entity-Based Knowledge Conflicts in Question Answering. In Marie-Francine Moens, Xuanjing Huang, Lucia Specia, and Scott Wen-tau Yih, editors, *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 7052–7063, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics.
- [LPP<sup>+</sup>21] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks, 2021.
- [LZB<sup>+</sup>22] Tianyu Liu, Yizhe Zhang, Chris Brockett, Yi Mao, Zhifang Sui, Weizhu Chen, and Bill Dolan. A Token-level Reference-free Hallucination Detection Benchmark for Free-form Text Generation, 2022.
- [MBK<sup>+</sup>22] Ayodele Marvellous, Anwasha Banerjee, Mark Kaplan, Alan Willie, Ria Kalluri, William Agnew, and Vinash Chouhan. Fine-tuning Large Language Models for Domain-Specific Legal Documents. 09 2022.
- [MFP<sup>+</sup>25] Nestor Maslej, Loredana Fattorini, Raymond Perrault, Yolanda Gil, Vanessa Parli, Njenga Kariuki, Emily Capstick, Anka Reuel, Erik Brynjolfsson, John Etchemendy, Katrina Ligett, Terah Lyons, James Manyika, Juan Carlos Niebles, Yoav Shoham, Russell Wald, Tobi Walsh, Armin Hamrah, Lapo Santarlaschi, Julia Betts Lotufo, Alexandra Rome, Andrew Shi, and Sukrut Oak. Artificial Intelligence Index Report 2025. Technical Report 8th edition,

Stanford Institute for Human-Centered Artificial Intelligence (HAI), April 2025.

- [MKL<sup>+</sup>23] Sewon Min, Kalpesh Krishna, Xinxi Lyu, Mike Lewis, Wen tau Yih, Pang Wei Koh, Mohit Iyyer, Luke Zettlemoyer, and Hannaneh Hajishirzi. FActScore: Fine-grained Atomic Evaluation of Factual Precision in Long Form Text Generation, 2023.
- [MLG23] Potsawee Manakul, Adian Liusie, and Mark J. F. Gales. SelfCheckGPT: Zero-Resource Black-Box Hallucination Detection for Generative Large Language Models, 2023.
- [MMN<sup>+</sup>25] Shervin Minaee, Tomas Mikolov, Narjes Nikzad, Meysam Chenaghlu, Richard Socher, Xavier Amatriain, and Jianfeng Gao. Large Language Models: A Survey, 2025.
- [P13] Björn Plüster. LEOLM: Igniting German-Language LLM Research. <https://laion.ai/blog/leo-lm>, Sep 28 2023. Accessed: 2025-09-01.
- [PXNO23] Ciyuan Peng, Feng Xia, Mehdi Naseriparsa, and Francesco Osborne. Knowledge Graphs: Opportunities and Challenges, 2023.
- [PZKS24] Venkatesh Balavadhani Parthasarathy, Ahtsham Zafar, Aafaq Khan, and Arsalan Shahid. The Ultimate Guide to Fine-Tuning LLMs from Basics to Breakthroughs: An Exhaustive Review of Technologies, Research, Best Practices, Applied Research Challenges and Opportunities, 2024.
- [RAT25] Ethan M. Rudd, Christopher Andrews, and Philip Tully. A Practical Guide for Evaluating LLMs and LLM-Reliant Systems, 2025.
- [Sch23] Philipp Schmid. Introducing IGEL an instruction-tuned German large Language Model. <https://www.philschmid.de/introducing-igel>, Apr 4 2023. Accessed: 2025-09-01.
- [Sha51] C. E. Shannon. Prediction and Entropy of Printed English. *The Bell System Technical Journal*, 30(1):50–64, 1951.
- [SSS<sup>+</sup>25] Pranab Sahoo, Ayush Kumar Singh, Sriparna Saha, Vinija Jain, Samrat Mondal, and Aman Chadha. A Systematic Survey of Prompt Engineering in Large Language Models: Techniques and Applications, 2025.
- [Tea25] Gemma Team. Gemma 3 Technical Report, 2025.
- [TMPB23] Miles Turpin, Julian Michael, Ethan Perez, and Samuel R. Bowman. Language Models don't always say what they think: Unfaithful Explanations in Chain-of-Thought Prompting, 2023.

- [TZJ<sup>+</sup>24] S. M Towhidul Islam Tonmoy, S M Mehedi Zaman, Vinija Jain, Anku Rani, Vipula Rawte, Aman Chadha, and Amitava Das. A Comprehensive Survey of Hallucination Mitigation Techniques in Large Language Models, 2024.
- [UK 08] UK Government. Coexistence Agreement – Fact Sheet. GOV.UK, 2008. Accessed: 2025-08-09.
- [vdBP24] Tim vor der Brück and Marc Pouly. Estimating Text Similarity based on Semantic Concept Embeddings, 2024.
- [Ver25] Vereinigung österreichischer Richterinnen und Richter. Stufenbau der Rechtsordnung. Online publication, 2025. Accessed: 2025-08-16, URL: <https://richtervereinigung.at/justiz/rechtssystem/stufenbau-der-rechtsordnung/>.
- [VSP<sup>+</sup>23] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is All You Need, 2023.
- [WCD24] Nathaniel Weir, Peter Clark, and Benjamin Van Durme. NELLIE: A Neuro-Symbolic Inference Engine for Grounded, Compositional, and Explainable Reasoning, 2024.
- [WSV<sup>+</sup>24] Benoist Wolleb, Romain Silvestri, Giorgos Vernikos, Ljiljana Dolamic, and Andrei Popescu-Belis. Assessing the Importance of Frequency versus Compositionality for Subword-based Tokenization in NMT, 2024.
- [WWS<sup>+</sup>23] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. Chain-of-Thought Prompting Elicits Reasoning in Large Language Models, 2023.
- [XJK25] Ziwei Xu, Sanjay Jain, and Mohan Kankanhalli. Hallucination is Inevitable: An Innate Limitation of Large Language Models, 2025.
- [XZ25] Tong Xiao and Jingbo Zhu. Foundations of Large Language Models, 2025.
- [ZSH<sup>+</sup>23] Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc Le, and Ed Chi. Least-to-Most Prompting Enables Complex Reasoning in Large Language Models, 2023.