Requirements Satisfiability with In-Context Learning

Sarah Santos, Travis Breaux
School of Computer Science
Carnegie Mellon University
Pittsburgh, Pennsylvannia, United States

Thomas Norton
School of Law
Fordham University
New York, New York, United States

Sara Haghighi, Sepideh Ghanavati
School of Computing and Information
University of Maine
Orono, Maine, United States

***Abstract*—Language models that can learn a task at inference time, called in-context learning (ICL), show increasing promise in natural language inference tasks. In ICL, a model user constructs a prompt to describe a task with a natural language instruction and zero or more examples, called demonstrations. The prompt is then input to the language model to generate a completion. In this paper, we apply ICL to the design and evaluation of satisfaction arguments, which describe how a requirement is satisfied by a system specification and associated domain knowledge. The approach builds on three prompt design patterns, including augmented generation, prompt tuning, and chain-of-thought prompting, and is evaluated on a privacy problem to check whether a mobile app scenario and associated design description satisfies eight consent requirements from the EU General Data Protection Regulation (GDPR). The overall results show that GPT-4 can be used to verify requirements satisfaction with 97.4% accuracy and dissatisfaction with 92.7% accuracy. Inverting the requirement improves verification of dissatisfaction to 96.7%. Chain-of-thought prompting improves overall GPT-3.5 performance by 9.4% accuracy. We discuss the trade-offs among templates, models and prompt strategies and provide a detailed analysis of the generated specifications to inform how the approach can be applied in practice.**

***Index Terms*—requirements, satisfaction arguments, language models**

I. Introduction

In software engineering, requirements serve to ensure that software is fit for purpose and designed to meet business objectives, legal requirements and stakeholder needs. In their seminal work, Zave and Jackson introduced satisfaction arguments, in which a system's specifications S and domain knowledge or properties K together satisfy the system requirements R, written as $S, K \vdash R$. This formulation has been used to motivate goal modeling refinement [39], means-end [25], [27] relationships and traceability [29] links, e.g., to show that sub-goals, which refine a parent goal, and domain properties are necessary and sufficient to satisfy the parent goal.

Satisfaction arguments expressed in logic have the benefit of reducing system behavior, domain properties, and requirements into discrete elements that can be analyzed and evaluated formally. However, these elements can lose the rich description of the argument and associated interpretability that underpins how the requirement is satisfied, and some requirements are difficult to express formally [33]. Maiden et al. [27] and Haley et al. [14] extend the formal argument with an informal natural language argument, which Haley et al. call the *inner argument*.

Recent advances in auto-regressive, large language models (LMs) have produced *in-context learning* (ICL), which is the model's ability to recognize a desired task and learn from analogy, often using examples at inference time [5]. In addition, LMs have been used to perform natural language inference (NLI), including open- and closed-domain reasoning and commonsense reasoning [32]. Improvements in NLI have been obtained using chain-of-thought (CoT) prompting, in which the LM is provided one or more reasoning steps prior to generating the answer [44].

In this paper, we study LM applications to generate and evaluate satisfaction arguments. We envision two use cases: (1) a software designer seeks to brainstorm which software features can be used to satisfy a requirement; and (2) given a design description, a designer wishes to check whether that description satisfies a requirement, and further to understand why.

The remainder of this paper is organized as follows: we introduce related work in Section II; in Sections III and IV, we present our approach and evaluation method; in Section V we present results with discussion in Section VI; in Section VII we discuss threats to validity; and we conclude in Section VIII.

II. Background

We now review related work on natural language inference, in-context learning, satisfaction arguments.

A. Natural Language Inference

Natural language inference (NLI), which includes Recognizing Textual Entailment (RTE) tasks, is the class of tasks for "determining whether a natural-language hypothesis can be inferred from a given premise" [28]. Given the text "A soccer game with multiple males playing," a model should confirm that "some men are playing a sport" is a valid hypothesis, and that "the woman kicked the ball" is a contradiction. Two prominent datasets exist to develop models for NLI, including the Stanford Natural Language Inference (SNLI) corpus [4], and Multi-Genre NLI (MultiNLI) corpus [46]. The SNLI contains 570K English sentence pairs labeled as entailment, contradiction, and neutral. MultiNLI contains 430K pairs using the same labels, but draws from a wider diversity of domains, including both spoken and written texts. The extended dataset e-SNLI includes explanations to support the predicted label [7]. At this time, state-of-the-art neural models exhibit a test accuracy of 93.1% on SNLI [40], and 90.8% on MultiNLI [23].

B. In-Context Learning

Recent advances in auto-regressive, large language models (LMs) have produced *in-context learning* (ICL), which is the model's ability to recognize a desired task and learn from analogy, often using examples at inference time [5]. For many benchmark natural language processing tasks, supervised deep learning has been the state-of-the-art, often requiring thousands and tens of thousands of training examples [24]. With ICL, LLMs have been shown to solve some benchmark tasks with no training examples, called *zero-shot learning*, and as few as 1-15 training examples, called *few-shot learning* [26], [50]. To achieve these results, a model user writes a natural language *prompt* consisting of an *instruction* that indicates the type of task, a text linearization called a *template* that includes trigger words and slots that are filled with expressions of human knowledge, and zero or more *demonstrations* or training examples that show the desired input and output [10].

The choice of which demonstrations to use and their ordering in ICL can have a significant effect on task accuracy by as much as 30% [26], [50]. In addition, LMs have been shown to exhibit a number of biases in classification tasks when using demonstrations: *majority label bias*, in which LMs choose the most common label among demonstrations; *recency bias*, in which LMs choose the most recent label from the last demonstration; and *common token bias*, in which LMs prefer to output tokens that are more common in their pre-training data [50]. These biases can be mitigated to some extent by "soft" prompt tuning, in which the prompt author chooses demonstrations with a near-equal balance of label classes and by ordering the demonstrations to evenly distribute labels.

An instance of ICL in which the demonstration exemplars include the question, the intermediary steps to complete before obtaining the answer, and the answer is called chain-of-thought (CoT) prompting [44]. Few-shot CoT prompting has shown improved performance in benchmark NLP tasks for arithmetic, symbolic and commonsense reasoning [18]. Zero-shot CoT prompting is used in the absence of exemplars using two-stage prompting: first, the model is prompted to generate the intermediary steps to reach an answer; next, the steps are amended to the first prompt to generate the answer. In arithmetic reasoning, zero-shot CoT performs better than zero-shot prompting, but performs 10% worse than few-shot prompting when using Google's PaLM 540B model. There are numerous variations on CoT, including plan-and-solve [42], self-consistency [41], interleaved CoT [38] and tree-of-thought prompting [47].

Instruction tuning is a fine-tuning procedure that improves LM performance when responding to human instructions in natural language [31]. The procedure uses training samples that fit into instructional categories, including brainstorming or making lists, generating narrative and text summarization, to name a few. Several commercial base models have been instruction tuned or provide instruction-tuned variants, including Gemini, Claude, GPT-3.5 and GPT-4. In addition to fine-tuning, prompt authors can divide the prompt template into regions using trigger words to improve performance [43]. Chain-of-thought prompting has been shown to improve performance in multi-hop reasoning tasks, wherein multiple facts must be reasoned about at once or when facts are inferred from other facts in series [44].

Large language models (LMs) are not impervious to failure. LMs have exhibited misdirection, which is the failure to follow instructions [31]. LMs have reported falsehoods, misinformation, and so-called hallucinations [22], and they have exhibited gender and racial bias toward others [30] and toxicity [13]. Moreover, they can exhibit *sycophancy* when they respond with the bias of their users, even if those biases are extreme or incorrect [45]. *Alignment* is a broader effort to improve LM performance and reduce these unwanted effects [1], and several commercial models, including Gemini, Claude, GPT-3.5, and GPT-4, have been fine-tuned to this end.

C. Satisfaction Arguments

Zave and Jackson first introduce *satisfaction arguments* using the formula $S, K \vdash R$, in which system specifications S and domain knowledge and properties K taken together must be sufficient to satisfy requirements R [49]. In goal modeling, a satisfaction argument is expressed as $SubGoals, DomProps \vdash ParentGoal$, in which satisfying the sub-goals and domain properties is sufficient to satisfy the parent goal [39]. In goal modeling notation, sub-goals are linked to a parent goal through a refinement relationship, and thus satisfaction arguments can be chained together to trace the satisfaction of an organization's business objectives down to a system's low-level requirements [39].

Maiden et al. [27] extended the $i*$ diagram notation by attaching satisfaction arguments to the means-end link in the notation. The argument is constructed from S, K and R (above) in addition to an informal, natural language argument justifying the satisfaction of R. Lockerbie et al. extended this work to reuse satisfaction arguments across socio-technical system boundaries [25].

In addition to refinement and means-end links, Murugesan et al. define traceability as a relationship between target and source artifacts [29]. The target artifacts Σ implement a system's behavior, such as code, and the source artifacts Δ are requirements. Thus, they write $S \vdash_r r$ for $S \subset Sigma$ and $r \in \Delta$ when S satisfies the requirement r.

Finally, Haley et al. structure satisfaction arguments in two parts: a formal *outer argument* expressed in logic that describes what system behavior entails the satisfaction of the requirement, e.g., $B \vdash R$, which means that requirement R is satisfied by the system behavior by the logical formula B; and an informal *inner argument* that supports the claims in the outer argument [14]. They propose using causal logic to express the outer argument and Toulmin-style arguments [37] to structure the outer argument.

III. Approach

We first introduce the applicaiton domain, before describing our approach to requirements satisfiability using in-context learning, which proceeds in three phases: (1) manually extract and summarize knowledge from authoritative sources for use in natural language inference; (2) generate specifications from public app descriptions; and (3) evaluate requirements satisfaction arguments using the summarized knowledge and generated specifications.

A. Application Domain

We made the following assumptions in choosing the application domain. (1) The domain knowledge should be authoritative, reusable, described in general terms independent of any one specification and, when combined with a specification, the knowledge should be sufficient to infer satisfaction. (2) While specifications can be represented formally (e.g., using logic or graph theory), we assume only informal, natural language descriptions of software. (3) The requirements are mathematically verifiable by satisfying a logical formula consisting of propositions. When true, each proposition corresponds to the satisfaction of an individual requirement described in natural language. (4) Arguments must justify why a requirement is or is not satisfied by drawing connections between the specification and the knowledge about the requirement in question.

Herein, specifications are presented as natural language scenarios with a list of design practices that describe how an application processes personal data. We ask whether the scenario satisfies eight requirements of consent under multiple articles of the EU General Data Protection Regulation (GDPR), including Articles 4(11), 6(1), and 7(4). The knowledge used to check satisfaction is derived from requirement-specific guidance [12] published by the European Data Protection Board (EDPB), which is the formal body in charge of ensuring the consistent application and enforcement of data processing law in the European Economic Area.

In addition to meeting our assumptions, this problem has additional constraints. While authoritative, the knowledge is limited and dated: the EDPB provides 30 pages of guideline description, including 24 examples, that were authored in May 2020 [12]. From this information, organizations must draw inferences about consent requirement satisfaction. Systems for computing satisfaction must be updated as new knowledge is discovered or created, either by new or amended guidelines, regulatory enforcement actions, or by legal cases. Presently, there are few legal cases or enforcement actions on this topic. We identified only one landmark legal case in Case C-252/21, where the Court of Justice of the European Union (CJEU) found that Meta Platforms violated the consent requirement of freely given by bundling unnecessary advertising practices with other platform data practices [6]. Similarly, there are few enforcement actions. In our review of 235 enforcement actions [1] decided between 30 June 2023 and 30 December 2023, we found only 50 cases that cover violations of consent articles, among which only 26% of these cases are likely judgments resulting from software design issues. Due to the limited authoritative ground truth, LMs offer an advantage in utilizing retrieval augmented generation [21] to incorporate authoritative regulatory guidance and chain-of-thought prompting [44] to make natural language inference explicit.

B. Knowledge

Knowledge about the domain and environment are needed to decide if a specification satisfies a requirement. We manually extracted the knowledge and requirements from the guidance document entitled “Guidelines 05/2020 on consent under Regulation 2016/679, Version 1.1” that was adopted on 4 May 2020. The document consists of 30 pages and 24 examples that describe scenarios in which systems satisfy or do not satisfy a given property.

We selected all eight requirements from Section 3 plus a ninth requirement from Section 5 of the guidance document to provide breadth in the kinds of phenomena covered. This includes requirements covering how consent is requested, the scope of data practices covered by consent, the scope of information provided to the data subject, and their access to withdrawal. We summarized the guidance in the following abbreviated rubric:[2]

- **Freely Given (F):** Consent is freely-given, if it exhibits all of the following: (1) is not presented to a data subject by a data controller with a power imbalance; (2) is not conditioned on accepting other terms, and not bundled; (3) is granular; and (4) yields no detriment.
  - **Power Imbalance (P):** Power imbalance generally occurs when the data controller is a public authority or employer, although other cases may arise. For a consent to be freely given in the presence of a power imbalance, the controller must demonstrate that there is no detriment when consent is refused or later withdrawn.
  - **Conditionality (C):** If the purpose for processing a data type is bundled with other contract terms, or if the data subject is otherwise compelled to consent, then it is conditional and is not freely given. Conditionality only applies if the requested data is unnecessary to perform the contract. Contracts include end-user agreements, terms of use, and terms and conditions.
  - **Granular (G):** Data subjects should be free to choose which purpose they accept, rather than having to consent to a bundle of processing purposes.
  - **Detriment (D):** The controller needs to demonstrate that it is possible to refuse or withdraw consent without detriment, including no deception, intimidation, coercion, or significant negative consequences. Gray Area: permissible incentives, which means a controller can use

[1]https://www.enforcementtracker.com/

[2]The complete rubric and source code will be made available in a non-anonymized replication package.

an incentive that is only obtainable if the data subject consents. This incentive is not viewed as a detriment to refusing to consent. Refusal to consent or withdrawal should not lead to a diminished product or service.

- **Specific (S):** The processing of data is limited to specific purposes and will not be processed for other purposes, the consent is granular, and the information presented to obtain consent describes the consent and not other unrelated matters.
- **Informed (I):** A design description must indicate that a data subject is informed prior to the collection of their data, and at minimum identify (a) the data controller's identity, (b) the purpose of each processing operation, (c) what type(s) of data will be collected and used, (d) the existence of the right to withdraw consent, (e) information about the use of the data for automated processing, and (f) about the risks due to transfers to countries without adequacy decisions or safeguards.
- **Unambiguous (U):** Consent must be provided through a clear, affirmative action, which may be a written, oral or electronic means.
- **Withdrawal (W):** The data subject can withdraw consent as easily as they gave it, and at any given time.

For each requirement definition in the rubric, we wrote a corresponding requirement statement in the optative mood. Requirements are generally written in the *optative mood*, which describes what we desire to be true, whereas the *indicative mood* describes what we assert to be true [16]. To check if a specification satisfies a requirement, we chose to write the requirements in the indicative mood, because this mood reflects what is or is not true of the specification, as opposed to what could be true, e.g., after modifying the specification or considering a hypothetical. The eight requirements are presented below:

P: There is a power imbalance between the data subject and the data controller.
C: The purpose for data processing is bundled with other contract terms, such as user agreements, terms of use, or terms and conditions.
G: The data subject can choose which data processing purposes they accept.
D: The data subject can refuse or withdraw consent and incur no detriment.
S: Data processing is limited to specific purposes.
I: The data subject is properly informed.
U: Consent is be provided through a clear, affirmative action by the data subject.
W: The data subject can withdraw consent as easily as they gave it.

The guidelines indicate that the freely given requirement F is a composition of four refinements, which we formalize in a sub-formula: $\neg P \wedge \neg C \wedge G \wedge \neg D$. Substituting this sub-formula for F, we observe that consent validity simplifies to: $\neg P \wedge \neg C \wedge G \wedge \neg D \wedge S \wedge I \wedge U \wedge E \wedge B \wedge W$. In the remainder of this paper, we refer to the eight non-refined requirements, which excludes freely given. Decomposing complex questions into more narrowly focused sub-questions is necessary to reduce the likelihood that the LM will miss relevant details during generation in the satsifiability task, and has been a performant prompting tactic in prior work [11], [17], [51].

We validated the above rubric by asking three investigators to decide whether the EDPB guidance examples satisfy each of the eight requirements. Among the 24 examples, six examples were excluded from this study because they were either redundant or did not describe a scenario with a target system (e.g., they restated content from recitals). Next, we sanitized the examples to remove any conclusory or justifying language indicating whether a requirement was satisfied and the rationale for satisfaction. For each of the requirements, coders were asked to assign one of the following codes: YES, if the scenario satisfies the property; NO, if the scenario does not satisfy the property; MIS, if the scenario is missing information needed to evaluate the property's satisfaction; or IDK, if the example contains relevant information, but the rubric is unclear. After each coder independently coded all 18 examples, Cohen's Kappa was 47%. Next, the coders met to discuss their disagreements, after which Kappa rose to 100%.

C. Specifications

As described in Section III-B, the number of authoritative specifications with ground truth labels is limited to 18 examples. To improve external validity, we use the LM to generate specifications from publicly available mobile application (app) descriptions. This has the advantage that the specifications are grounded in real-world applications and we can experimentally control the breadth and diversity of application behaviors. This step also simulates how developers can use LMs to "brainstorm" and discover design practices that do and do not satisfy requirements.

The generation process consists of a pipeline of LM tasks, in which the output of an upstream task becomes the input to a downstream task. To describe this process formally, let I be an instruction to an LM that indicates the task type, and let T be a template function that maps one or more text values $(v_1, v_2, ...v_n)$ to slots in a text linearization, yielding a slot-filled text t. A prompt is the concatenation of one or more strings, expressed using semi-colon, e.g., the prompt $P = I; T(v_1)$ for an instruction I and a text t based on the template T that had one slot filled by the value v_1.

For each task, we used LangChain v0.0.344 and the OpenAI API with the *gpt-3.5-turbo-1106* model and parameters $temperature = 0.7$, $top\_p = 1.0$. We chose this model for the lower pricing and larger context window of 16,385 tokens. The temperature and *top_p* parameters control token sampling during generation. Lower temperature gives preference to higher probability tokens, thus reducing randomness and increasing focus across generations, whereas higher temperature generally increases randomness. The *top_p* is an alternative to temperature that use nucleus sampling by choosing a subset of tokens with a cumulative probability mass above the *top_p* threshold. In our study, we effectively ignore nucleus sampling and chose a moderately lower temperature $0.0 < 0.7 < 2.0$.

We begin with the top 50 mobile application (app) descriptions reported separately by the Google Play and Apple App stores for each of the 27 EU member states. This initial set yields 2,482 unique Apple App apps and 1,559 unique Google Play apps. Next, we randomly selected 200 app descriptions from each app store dataset. The generation process continues through the following steps:

1) For each app description $a_i \in A$ for $0 < i \leq |A|$, summarize a_i into a one sentence summary z_i, using the prompt $P_1 = I_1; T_1(a_i)$
2) For each app description $a_i \in A$, extract a list of data practices D_i, using the prompt $P_2 = I_2; T_2(a_i)$
3) For each data practice $d_{i,j} \in D_i$, identify a list of candidate data types $t_{i,j}$ likely used by the practice, using the prompt $P_3 = I_3; T_3(d_{i,j})$
4) For each summary z_i, data practice $d_{i,j}$ and list of data types $t_{i,j}$ for this practice, write a brief scenario c_i using the prompt $P_4 = I_4; T_4(z_i, d_{i,j}, t_{i,j})$
5) For each scenario c_i, requirement $r_m \in R$ and domain knowledge k_m defining the requirement, choose one of two augmentations for c_i at random: a) assume r_m is true, then generate a list of satisfactory design practices using the prompt $P_5^+ = I_5^+; T_5(c_i, k_m, r_m)$; or b) assume r_m is false, then generate a list of dissatisfactory design practices using the prompt $P_5^- = I_5^-; T_5(c_i, k_m, r_m)$. For each list of design practices X_i generated using scenario c_i, let S_m^+ be the set of pairs (c, X), called a specification, that were generated using prompt P_5^+ to satisfy requirement r_m, and let S_m^- be the set of specifications generated using prompt P_5^- to not satisfy requirement r_m.

At step 3 above, the 200 app descriptions for Apple App and Google Play yielded 1,637 and 2,148 triples consisting of a shared summary, a single data practice, and data types associated with the practice, respectively. As inputs to generating app scenarios in step 4, we randomly sampled 200 triples for each app store. Step 4 yielded 400 scenarios in total. Below is an example scenario generated from steps 1-4:

> **Summary**: The mobile app is a platform for watching and creating short-form videos that are personalized to your interests, with easy-to-use tools for editing and adding effects, music, and filters.
> **Data Practice**: The user watches short-form videos on TikTok.
> **Data Types**: ['watch history', 'liked videos', 'comments', 'user profile information']
> **Scenario**: The app provides a user with a personalized video feed that offers a wide range of entertaining and inspiring short videos. It also offers simple-to-use tools for creating original videos with special effects, filters, and music. The user is able to edit their own videos using the app's features. The app utilizes videos to enable this functionality.

In step 5, we extend each scenario with a list of design practices generated by the model that collectively satisfy or dissatisfy one of the eight requirements described by the rubric in Section III-B. For an even distribution across app stores, requirements and satisfaction states, this yields 384 total specifications down-sampled from the 400 scenarios randomly sampled in step 4. Next, we generate design practices D using deduction expressed by the formula $K, S, R \vdash D$. Because LMs can be fine-tuned to follow instructions [31] and aligned to limit undesired behaviors [1] (e.g., toxicity [13], social bias [30] and hallucinations [22]), and because K can overlap with the pre-training data [5], which can cause related but irrelevant information to "leak" into the model output, we conducted a series of design iterations to shape the answers, called answer shaping [24]. Answer shaping includes changes to the input to exclude undesirable output and retain desirable output.

Below is an example list of design practices generated by step 5 for the consent requirement of specificity. Each specification consists of the design practice list appended to the end of the generated scenario.

1) The app clearly and specifically states the purposes for which the data will be processed, such as providing a personalized video feed and offering video creation tools with special effects, filters, and music. This ensures that the data processing is limited to these specific purposes.
2) When the user first installs the app, they are presented with a consent request that is granular and allows them to choose whether they consent to each specific purpose separately. For example, they can choose to consent to the personalized video feed but not to the video creation tools, or vice versa. This granular consent ensures that the user has a choice in relation to each specific purpose.
3) The information presented to obtain consent focuses solely on the specific purposes for which the data will be processed. The consent request does not include unrelated matters, such as promotional offers or unrelated data processing activities. This clear separation of information ensures that the consent request describes the consent and not other unrelated matters.
4) The app does not process the user's data for purposes other than those specifically stated. It does not use the data for targeted advertising or share it with third parties without the user's explicit consent. This adherence to purpose specification safeguards against function creep and ensures that the data processing remains limited to the specific purposes for which the user has given consent.

We evaluated the generated dataset in two steps. First, two investigators involved in designing the generation process manually reviewed the 400 sampled scenarios to validate that the data types are relevant to the data practices and that the practices are relevant to the summary. This step identified 19 scenarios for removal due to practices outside app scope, no practices, and misinterpretation of a fictional game. Next, one of these investigators reviewed the generated design practices for logical consistency with whether the practices were expected to satisfy the requirement. We report and discuss the results of this review in Section VI.

Finally, two investigators not involved in the generation process design used the rubric described in Section III-B to classify each generated specification by whether it satisfies or does not satisfy the given requirement and to record their justification for their classification. They first coded 16 specifications before meeting to discuss the process. At this stage, the initial inter-rater reliability calculated using Cohen's Kappa [9] was K = 50%, which is considered a moderate agreement [19]. Next, they coded the remaining specifications independently, yielding a Kappa K = 76%, which is considered substantial. The two investigators further discussed and re-coded five remaining disagreements to yield K = 91%, which is considered *almost perfect*.

Among the disagreements, most occur in specifications related to the "conditionality" and "power imbalance" requirements. In specification SCR-A003, the design practice list states the "Conditionality" to be false, as the data processing now requires the data subject (user) to accept terms and conditions." The first investigator decided that conditionality is true because the user must agree to the terms and conditions before using the app, while the second investigator agreed with the quoted conclusory statement and marked conditionality as false. To mitigate these disagreements, the investigators agreed to ignore the conclusory statements and make their own judgement.

The final dataset consisting of the reconciled disagreements were used as the ground truth to evaluate the requirements satisfiability phase, which we now discuss.

D. Satisfiability

We check whether a specification s satisfies a requirement r_m using knowledge k_m specific to the requirement and the prompt $P_6 = I_6; T_6(s, k_m, r_m)$. When checking satisfaction, we retrieve the knowledge and requirement from a database, which can be updated over time. The knowledge k is comprised on a requirement name and definition of that requirement, and the specification $s = (c, X)$ is comprised of the app scenario c and list of design practices X for the given app. To satisfy a requirement r, it must be true that no design practice $x \in X$ dissatisfies r; and to dissatisfy r, it must be true that at least one $x \in X$ dissatisfies r. To test whether LMs are robust reasoners when requirements statements are inverted, we check satisfiability separately using the requirement r_m and the inverted requirement, written r_m^-. For example, the requirement for specificity is written as "data processing is limited to specific purposes" and the inversion is written as "data processing is not limited to specific purposes." Therefore, we can check for consistency such that $s, k \vdash r$ is not true, if and only if $s, k \vdash \neg r$ is true.

1) Instructions and Templates: Prompt performance has been shown to vary widely based on the choice of words in the instruction and template [36], [50]. This variance is attributed to the word distributions learned during the model pre-training and to how in-context learning uses token sequences to select the task at inference time. Thus, we examine two different templates by varying the prompt vocabulary. In Listing 1, we present a *requirements template* T_6^R that uses an instruction written about requirements satisfiability and that uses trigger words drawn from this vocabulary (e.g., specification and requirement). In this template, the answer is missing and the model is instructed to complete the answer with True or False. In Listing 2, we present a *generic template* T_6^G that is written about a scenario and statement that must be true of the scenario. Apart from these changes in the trigger words and in how to choose a response, the two templates are identical.

Listing 1: Satisfiability Template R

```
Definition of {req_name}: {definition}

Read the following specification. If the
    specification satisifies the requirement
    based on the definition, above, respond
    with True, otherwise respond with False.
    Do not comment or elaborate.

Specification: {scenario} {design_practices}

Requirement: {requirement}

Answer:
```

Listing 2: Satisfiability Template G

```
Definition of {req_name}: {definition}

Read the following scenario, and decide if the
    given statement is true or false based on
    the definition, above. Respond with True
    or False. Do not comment or elaborate.

Scenario: {scenario} {design_practices}

Statement: {requirement}

Answer:
```

2) Chain-of-Thought Prompting: Prompts that involve reasoning over multiple hops or facts have shown improved performance using Chain-of-Thought (CoT) prompting in natural language inference tasks [44]. In CoT prompting, a prompt is augmented with one or more examples that demonstrate step-by-step reasoning. The model then completes a reasoning task prior to providing an answer to a given question. The key idea is that the generated reasoning reduces the probability of misdirection when completing the answer. To further narrow the focus in this task, we chose to apply CoT directly to the list of design practices and to exclude the scenario. In Listing 3, we introduce prompt $P_7 = I_7; T_7(X, k_m, r_m, E)$ that checks whether a list of design practices X satisfy requirement r_m given knowledge k_m. Based on an early evaluation of P_6, we based this template on the generic template T_6^G. The template T_7 embeds a list of training examples E in which each example $e \in E$ consists of a list of design practices, a requirement, an answer to whether the design practices satisfy the requirement, and a rationale justifying the answer (see sub-template T_7^E in Listing 4). To separate the examples, we use the trigger word # END, and to separate the question and answer components we use the trigger word ###. These examples are drawn from the ground truth dataset and kept separate from the data used for evaluation.

Listing 3: Chain-of-Thought Template

```
Definition of {req_name}: {definition}

Read the following example scenarios and
    observe the rationale and answer about
    whether the statement is true or false.
    For the last scenario and statement,
    decide if the statement is true or false
    based on the definition, above. Respond by
    completing the Rationale and Answer using
    the same format. Do not elaborate.

{examples*}

Scenario: {design_practices}

Statement: {requirement}

###

Rationale:
```

In Listing 3, the {`examples*`} slot is filled with one or more examples generated by the sub-template show in Listing 4. The slot fillers in each sub-template are drawn from the ground truth data hold-out dedicated to training.

Listing 4: Chain-of-Thought Sub-Template

```
Scenario: {design_practices}

Statement: {requirement}

###

Rationale: {rationale}

Answer: {answer}

# END
```

As described in Section II, the choice of which demonstrations to use and their ordering can impact accuracy by as much as 30% [26], [50]. This is due in part to *majority label bias*, in which LMs choose the most common label among demonstrations, and *recency bias*, in which LMs choose the most recent label from the last demonstration [50]. These effects are reduced as model size increases. Regardless, to mitigate any possible effects of these biases, we divide the training examples into subset E_m^+ for those examples that demonstrate when requirement r_m is satisfied, and into subset E_m^- for those examples that demonstrate when requirement r_m is not satisfied. Next, we check the satisfiability of r_m for a previously unseen list of design practices X by presenting demonstrations in the ordering $e_0^+; e_0^-; ...; e_n^+; e_n^-$ for $e_i^+ \in E_m^+$ and $e_i^- \in E_m^-$ and $0 < i \leq min(|E_m^+|, |E_m^-|)$. This ordering ensures an equal number of examples from each class (satisfies and does not satisfy) and distributes the classes evenly in the order. Finally, if we are testing an inverted requirement, then we invert the requirement and answer in all of the selected examples.

IV. Evaluation

We evaluate the efficacy of checking requirements satisfiability by the proportion of correct LM responses (c) to the total LM responses (t), which is equal to c/t, called *accuracy*. To check if a response is correct, we first check if the case-insensitive response is in the set `(True, False)`, which we call a *uniform response*. This check is necessary because generative LMs can produce answers that are not limited to a discrete set, despite considerable answer shaping. If the response is not `True` or `False`, which we call a *non-uniform response*, we next try to match the response to the regular expression `/Answer: (True|False)/` based on the template design and trigger word `Answer`, and to the expression `/The (requirement|statement) ("?.+?"?\s)?is (true|false)/`, which is a common elaboration featured by the model that in some cases includes the requirement restated in quotes. This approach is helpful to detect correct responses when the LM generates additional commentary and embeds the answer in this commentary. If the regular expressions do not match a response, which we call a *non-parsable response*, we count this response as incorrect. If the expression matches one or more times, then we accept the last match as the predicted answer and check the predicted answer against the expected answer in the ground truth dataset. We report the number of non-uniform responses that do not match the above patterns for each experiment in Section V.

In addition to computing overall accuracy, we are interested in how accuracy varies by requirement type, how the effects of the polarity of the scenario affect accuracy, which is whether the scenario satisfies or dissatisfies the requirement, and how the polarity of the requirement affects accuracy, which is whether the requirement is written in terms of what the system does or does not do. Finally, we examine whether running a prompt 10 times and taking the majority response (vote) affects accuracy, which is called self-consistency [41] and has shown promise in prior work [20]. To this end, we conduct experiments to answer the following research questions:

RQ1: How does accuracy vary by the requirement type?
RQ2: How does the scenario polarity affect accuracy?
RQ3: How does the requirement polarity affect accuracy?
RQ4: How does majority response affect accuracy?

To evaluate overall accuracy and answer the research questions, we used the ground truth dataset described in Section III-C. We sampled the dataset to ensure half of the specifications were generated from the 1,637 Apple App scenarios, and the other half from the 2,148 Google Play scenarios. In addition, we sampled to ensure an even distribution across the eight requirements and two satisfaction states, wherein a specification either satisfies or does not satisfy one of the eight requirements. We restricted the sample total to less than 400 specifications, which yields 384 specifications in which each requirement and satisfaction state was replicated 24 times, and each specification represents a different mobile application (i.e., $8 \times 2 \times 24 = 384$). Because Chain-of-Thought prompting requires training examples, we held out 20% of the ground truth dataset based on a near even distribution across all eight requirements, which yields 75 training samples, and 300 testing samples. We evaluated each satisfiability prompt P_6 and P_7 using the same 300 testing samples, which were evenly distributed across requirements.

In-context learning relies on token sampling to generate a model completion. Sampling is a non-deterministic process that can be affected by changing the model $temperature$ or top_p parameters. This can lead to different responses, even when using the same prompt. Therefore, we checked satisfiability by prompting the model 10 times for each requirement and its inversion to yield a total of $300 \times 2 \times 10 = 6,000$ LM responses per experiment. We report the mean accuracy of the 10 trials for each experiment. For the majority response, we calculate the frequencies for a True and False response, respectively, and choose the response with the highest frequency. If the responses are tied, we randomly choose True or False as the response.

For this task, we used LangChain v0.0.344 and OpenAI API with the *gpt-3.5-turbo-1106* model, which has a 16,385 token context window, and the *gpt-4-0613* model, which has a 8,192 token window. Both models have a pre-training cut-off date of September 2021. In each experiment, we use the same parameters $temperature = 0.7$, $top\_p = 1.0$.

V. Results

We now discuss the results based on the approach described in Section III. In Table I, we present the mean accuracies for the 10 trials for each experiment and majority response described in the evaluation method in Section IV. The columns correspond to each experiment, including the requirements T_6^R and generic T_6^G templates, the best performant 1-shot Chain-of-Thought (CoT) template T_7 and both GPT-3.5 and GPT-4 models. The rows present the per-requirement mean accuracy, the accuracy when the specification satisfies the requirement (Spec. true) and dissatisfies the requirement (Spec. false), and when the requirement was written in the normative and inverted tone, the overall accuracy followed by the accuracy when taking the majority response from 10 prompt responses. The highest accuracy for each experiment is presented in **bold**. All experiments used the same dataset for this evaluation.

	GPT-3.5			**GPT-4**	
	T_6^R	T_6^G	T_7	T_6^R	T_6^G
Power Imbalance	0.878	0.895	0.896	**0.961**	0.934
Conditionality	0.695	0.767	0.828	0.861	0.858
Granularity	0.613	0.851	**0.999**	0.986	**0.999**
Detriment	0.849	0.879	0.962	**1.000**	**1.000**
Specificity	0.724	0.820	0.949	**0.978**	0.974
Informed	0.646	0.807	0.992	**1.000**	0.982
Unambiguous	0.756	0.783	0.910	0.953	**0.961**
Withdrawal	0.769	0.927	0.946	0.950	0.946
Spec. (true)	0.841	0.883	0.954	0.974	**0.977**
Spec. (false)	0.478	0.731	0.886	**0.927**	0.902
Req.	0.544	0.852	0.940	0.955	**0.959**
Req. (inverted)	0.938	0.831	0.930	**0.967**	0.954
Overall Accuracy	0.741	0.841	0.935	**0.961**	0.957
Maj. Response	0.742	0.870	0.948	**0.963**	0.957

TABLE I: Mean Accuracy over 10 Experimental Trials

The highest overall accuracy for checking requirements satisfaction was 95.7% using GPT-4 and the requirements template T_6^G, followed by GPT-4 with the generic template and then GPT3.5 with 1-shot Chain-of-Thought (CoT) prompting. CoT has the advantage that responses include the rationale or justification for the answer, despite the lower accuracy. In these experiments, we only observed unparsable, non-uniform responses in the CoT experiment that accounts for 0.002% of the error in that reported result.

RQ1 asks "How does accuracy vary by the requirement type?" To answer RQ1, we examine the accuracy for each requirement. In Table I, Conditionality was overall the weakest performing satisfiability check across all models and prompt types. Conditionality was also a requirement that the human evaluators identified the most disagreements over. In weaker performing experiments, Informed is a close second in weakest performance, whereas GPT-4 performs very well for this requirement. Notably, CoT outperforms all other models and templates for Informed reaching 99.2% accuracy.

The question RQ2 asks "How does the scenario polarity affect accuracy?" Within the ground truth dataset, 192 specifications were generated to satisfy a requirement (true), and 192 specifications were generated to dissatisfy a requirement (false). In general, we observe in Table I that accuracy falls when the specification does not satisfy the requirement. The highest performance loss is GPT-3.5 with the requirements template T_6^R, dropping to 47.8% accuracy as compared to 73.1% with the generic template. Even the best performing GPT-4 experiments exhibit a 5-8% difference in performance.

The question RQ3 asks "How does the requirement polarity affect accuracy?" In this experiment, the requirement was written as a normative statement describing *what a system should do*, and as an inverted statement describing *what the system should not do*. The requirement template, the normative statement and GPT-3.5 shows the lowest accuracy (54.5%) with less difference among GPT-3.5 CoT and GPT-4.

Finally, RQ4 asks "How does the majority response affect accuracy?" In this experiment, we observe that majority response generally yields an accuracy close to or slightly above (2-5%) the mean. This observation means this option is preferred, since there are trials among these experiments where the accuracy is below the the mean.

In Table I, we presented the best n-shot CoT experiment, which is $n = 1$. In Figure 1, we present the overall mean accuracy for the GPT-3.5 generic template T_6^G experiment to represent $n = 0$ and the CoT template (T_7) experiments for $n = 1, 2, 4, 8$. The one-shot experiment yields the highest overall accuracy and declines up to four-shots before increasing again with eight-shot. We discuss an explanation for this decline in Section VI after performing an error analysis on the unparsable, non-uniform responses.

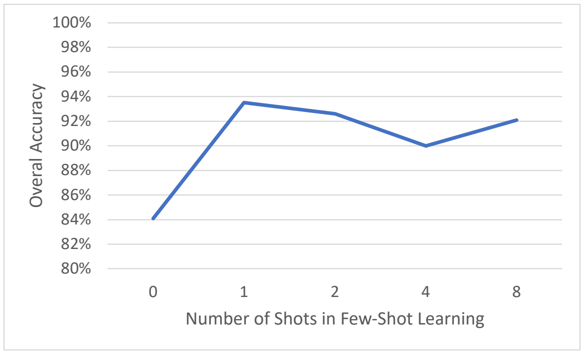

Fig. 1: n-Shot CoT Accuracy with Generic Template

Ans.			**GPT-3.5**				**GPT-4**	
	T_6^R	T_6^G	$T_7$1S	$T_7$2S	$T_7$4S	$T_7$8S	T_6^R	T_6^G
0	0	0	11	14	45	37	0	0
1	14	8	5918	5806	5776	5746	0	0
2	0	0	69	158	115	131	0	0
3	0	0	0	20	4	4	0	0
4+	0	0	0	0	59	13	0	0

TABLE II: Non-uniform Responses among 6000 Total

Table II presents frequencies of non-uniform responses for each experiment. The first column indicates the number of parsable answers, with the remaining columns corresponding to the experiments with each of the requirements (T_6^R) and generic templates (T_6^G) tested, the n-shot (nS) CoT template (T_7) and the two models, GPT-3.5 and GPT-4. The rows correspond to the number of responses out of the total of 6,000 possible responses for each experiment. Nearly all of the CoT template responses were non-uniform and 4-shot CoT showing the highest number of non-parsable responses. The generic template used with GPT-4 shows zero non-uniform responses.

VI. Discussion

We now discuss and interpret our results.

While 1-shot Chain-of-Thought (CoT) prompting shows improvement over non-CoT prompting with GPT-3.5, we observed a decreasing accuracy with increasing n and a high number of non-uniform responses. The pattern with non-uniform responses shown in Table II for 1-shot CoT is consistent across all n for $n = 1, 2, 4, 8$: most responses are non-uniform, however, the number of unparsable non-uniform responses where zero answers were received increases in proportion to the loss of accuracy. We examined the unparsable responses and observed the model was increasingly ignoring the instructions with phrases such as "Apologies, but I cannot fulfill that request." and in one case "I'm sorry, but I cannot complete this task as it requires making judgments about legal and ethical matters." In 4-shot CoT where overall accuracy was the least, 33/45 unparsable responses exhibit this pattern.

We attribute these unparsable responses to the practice of *alignment*, which aims to produce models that are helpful, honest and harmless [1]. Alignment is to reduce toxicity in responses and to prevent providing responses that conflict or compete with licensed professional advice, such as legal or medical guidance. Because our chosen application domain is legal compliance, the CoT prompts may have triggered alignment protocols. The models that we studied, *gpt-3.5-turbo-1106* and *gpt-4-0613*, have both been instruction-tuned and aligned according to OpenAI's public disclosures. An alternative to using an aligned model is to use a base model, such as LLaMA2 or *davinci-002*, that has not been instruction tuned nor aligned. The downside is that instruction tuned models have shown improved performance over base models on benchmark NLP tasks [43]. That said, aligned models exhibit an "alignment tax" that reduces NLP performance [31].

We conducted an analysis of the total 3,291 generated design practices across 384 specifications to identify errors and other issues in the generation method described in Section III-C. Overall, we observed 29 specifications (7.55%) in which the design practices exhibit the opposite polarity of the intended requirement. Nine of these specifications were generated to fit Conditionality, and the remaining 20 to fit Detriment. We observed 12 specifications with a definition bias (3.13%), in which certain elements of a definition are emphasized while others are minimized or missing. For example, the practices for Conditionality overemphasize "accepting terms and conditions" and under-emphasize "bundling consent with unnecessary data."

The generated design practices often contained conclusory statements that explicitly refer to whether the requirement is satisfied. We identified 256 specifications (66.67%) with conclusory statements, among which 66 could be overtly leading. For example, SCR-G028 states "...the user is consenting to the processing of their personal data for multiple purposes related to enhancing the security of their online accounts. However, since the user cannot choose which specific processing purposes they accept or give separate consent for each purpose, the 'granularity' requirement is not fulfilled." Conclusory statements also appear in the EDPB authoritative examples, and were ignored (if they were inaccurate) when creating the ground truth dataset.

We observed that 3.91% of specifications contained irrelevant practices with a conclusory statement, e.g., in SCR-G124 "These actions do not directly address the power imbalance between the data subject and the data controller. However, it is important to note that the presence of power imbalance is not negated by these design practices."Six specifications (1.56%) exhibited fallacious reasoning, including practice #5 of SCR-G144 "The user saves the collage, indicating that the data processing was limited to the specific purpose of creating the collage and not used for other unrelated matters." This is unsound, because "saves the collage" does not entail "processing was limited to the specific purpose."

We observed a few instances logical inconsistency. Five specifications (1.30%) contain logically inconsistent design practices. For example, in SCR-G187, design practice #5 states "The user reads and accepts the terms and conditions," while practice #13 states "The user chooses not to accept the terms and conditions."Finally, we observed three specifications

(0.78%) with double negatives, including SCR-G080 "This design practice does not cause 'Detriment' to be false."

VII. Threats to Validity

We now discuss threats to validity.

Construct validity is the correctness of operational measures used to collect data, build theory and report findings from the data [48], and the extent to which an observed measurement fits a theoretical construct [35]. To reduce this threat, we developed the knowledge and rubric described in Section III-B used to generate the design practices and evaluate satisfiability from the authoritative European Data Protection Board consent guidelines [12]. In addition, we divided the investigators into the process group and evaluation group. The process group evaluated the rubric using the 18 examples described in the guidelines, and the evaluation group evaluated the specifications generated using the rubric. Finally, we seeded every specification using mobile app summaries and data practices extracted from real descriptions of top, most popular apps used in the jurisdiction under which the legal requirements apply.

It is likely that the knowledge used to generate the design practices is incomplete, and that other unforeseen design practices will arise as technology evolves. At present, industry only has 18 authoritative examples from which to evaluate their own designs, and there are few regulatory enforcement actions and few judicial cases to clarify unforeseen situations. Thus, future work should study the effects of extending the knowledge to address new challenges.

In addition, the design practices themselves were generated by the LM in zero-shot setting with only the knowledge to guide the generation. This step yields synthetic data that may not be representative of actual design practices. To mitigate this threat, the process group reviewed the generated design practices for anomalies and inconsistencies. However, they did not evaluate the generated practices using outside knowledge, e.g., using a survey of industrial practices.

Internal validity is the extent to which measured variables cause observable effects in the data [48]. In this study, we created synthetic data consisting of design practices generated by the GPT-3.5 language model using knowledge provided by an authoritative source and the model's pre-training data. In two experiments, we used the same model to check whether the generated specifications satisfy legal requirements described in the same source of knowledge. It's possible that the model's output in the experiments was biased by the model's output in the specification generation step to predict the expected answer. To mitigate this step, we had two investigators not involved in the generation process to independently create expected answers in a ground truth dataset. This dataset was used in the evaluation to ensure independence between generation and satisfiability checking. In addition, we employed a second larger model GPT-4 to evaluate results independently from the model used to generate the specifications.

It is possible that the EDPB guidelines that were used to create the knowledge base, including the 18 examples, were part of the LM pre-training data. The cut-off dates for the GPT-3.5 model *gpt-3.5-turbo-1106* and GPT-4 model *gpt-4-0613* are September 2021. This means performance may worsen or require additional context in cases where the requirements are completely unseen by a different LM.

External validity determines the scope of environmental phenomena or domain boundaries to which the theory and findings generalize [48]. In this study, we examined requirements in a legal domain applied to mobile applications. The requirements cover a range of system behavior described in the specifications, including obligations of the app developer and their firm, how data is processed, how consent is collected and what happens when consent is withdrawn, and how information is presented to app users. Other phenomena not covered by this setting include performance requirements, including concurrency and parallelism in computation, safety critical requirements, and security requirements.

In addition, prior research has shown that prompt performance does not transfer between models in a predictable way [26]. This is true across different size models of the same generation (e.g., 13B versus 175B GPT-3). In our experience, transferabilty is increasingly difficult as models undergo different fine-tuning practices (e.g., instruction-tuning [43], alignment [31] and function calling [34]).

VIII. Conclusion and Future Work

In this paper, we report on experiments to check specifications for requirements satisfiability using large language models and in-context learning. We studied eight requirements from the legal domain using two popular models, GPT-3.5 and GPT-4. The experiments were conducted using synthetic data generated from popular mobile app descriptions that we acquired from jurisdictions regulated by this domain. In addition to different prompt templates with and without chain-of-thought (CoT) prompting, we studied LM performance when the specification both satisfies and dissatisfies the requirement and when the requirement is written in a normative and inverted tone. The results indicate that a generic prompt template outperforms a requirements theory-specific template, that CoT prompting improves mean accuracy well above non-CoT prompting with GPT-3.5, and that GPT-4 outperforms all other approaches studied.

The research highlights a number of challenges for future work. First, we see a need for more work to studying relationship between natural language and logical inference and to understand how these two approaches could complement one another. Because knowledge evolves, we need new tools to critique existing satisfaction arguments that no longer hold under changing requirements definitions. Finally, we see the ability to generate specifications (i.e., generative requirements engineering) as a rich opportunity for design space exploration. However, we need methods to engage humans in the analysis and comprehension of generative RE that can leverage and build on emerging work in reinforcement learning with human feedback (RL4HF) [8].

Acknowledgment

This research was funded by NSF Award #2007298, #2217572.

References

[1] A. Askell, Y. Bai, A. Chen, et al. "A General Language Assistant as a Laboratory for Alignment," arXiv:2112.00861, 2021

[2] K. Attwood, T. Kelly, J. McDermid, "The Use of Satisfaction Arguments for Traceability in Requirements Reuse for System Families: Position Paper," *International Workshop on Requirements Reuse in System Family Engineering*, 2004.

[3] R. A. Bauer, "Consumer behavior as risk taking," R*isk Taking and Information Handling in Consumer Behavior* 1960, pp. 389-398.

[4] S.R. Bowman, G. Angeli, C. Potts, C.D. Manning. "A large annotated corpus for learning natural language inference." *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2015.

[5] Brown et al., "Language Models are Few-Shot Learners," *Advances in Neural Information Processing Systems (NeurIPS)*, 33, 2020.

[6] "Judgment of the Court of Justice in Case C-252/21 — Meta Platforms et al. (General Terms of Use of a Social Network)." (in German) Press Release No. 113/23, Luxembourg, 4 July 2023.

[7] O-M. Camburu, T. Rocktäschel, T. Lukasiewicz, P. Blunsom. "e-SNLI: Natural Language Inference with Natural Language Explanations," *Advances in Neural Information Processing Systems (NeurIPS)* v. 31, 2018.

[8] P.F. Christiano, J. Leike, T. Brown, M. Martic, S. Legg, D. Amodei. "Deep reinforcement learning from human preferences." *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

[9] J. Cohen. "A coefficient of agreement for nominal scales," *Educational and Psychological Measurement*, 20: 37-46, 1960

[10] Q. Dong, L. Li, D. Dai, C. Zheng, Z. Wu, B. Chang, X. Sun, J. Xu, L. Li and Z. Sui, "Survey on In-context Learning," arXiv:2301.00234

[11] D. Dua, S. Gupta, S. Singh, M. Gardner. "Successive Prompting for Decomposing Complex Questions." *Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1251–1265, 2022.

[12] European Data Protection Board, "Guidelines 05/2020 on consent under Regulation 2016/679," Version 1.1, adopted 4 May 2020.

[13] S. Gehman, S. Gururangan, M. Sap, Y. Choi, N.A. Smith. "RealToxicityPrompts: Evaluating Neural Toxic Degeneration in Language Models." *Findings of the Association for Computational Linguistics*, pp. 3356–3369, 2020.

[14] C.B. Haley, R. Laney, J.D. Moffett, B. Nuseibeh, "Arguing satisfaction of security requirements." In: H. Mouratidis, P. Giorgini, eds. *Integrating security and software engineering: advances and future vision.* IG Press, 2006.

[15] C. Haley, R. Laney, J. Moffett and B. Nuseibeh, "Security Requirements Engineering: A Framework for Representation and Analysis," *IEEE Transactions on Software Engineering*, 34(1):133-153, 2008

[16] M. Jackson, "The World and the Machine," *International Conference on Software Engineering (ICSE)*, pp. 283-283, 1995.

[17] T. Khot, H. Trivedi, M. Finlayson, Y. Fu, K. Richardson, P. Clark, A. Sabharwal, "Decomposed Prompting: A Modular Approach for Solving Complex Tasks" *International Conference on Learning Representations*, 2023.

[18] T. Kojima, S. Gu, M. Reid, Y. Matsuo, Y. Iwasawa. "Large Language Models are Zero-shot Reasoners." *Advances in Neural Information Processing Systems (NeurIPS)* 35, pp. 22199-22213, 2022.

[19] J.R. Landis, G.G. Koch. "The measurement of observer agreement for categorical data." *Biometrics* 1977, 33: 159-74.

[20] B. Lester, R. Al-Rfou, N. Constant. "The Power of Scale for Parameter-Efficient Prompt Tuning," *Empirical Methods in Natural Language Processing (EMNLP)*, pp. 3045–3059, 2021.

[21] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W. Yih, T. Rocktäschel, S. Riedel, D. Kiela. "Retrieval-augmented generation for knowledge-intensive NLP tasks." *Advances in Neural Information Processing Systems* (NeurIPS), 2020.

[22] S. Lin, J. Hilton, O. Evans. "TruthfulQA: Measuring How Models Mimic Human Falsehoods." *Association for Computational Linguistics (ACL)*, pp. 3214–3252, 2022

[23] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, V. Stoyanov, "RoBERTa: A Robustly Optimized BERT Pretraining Approach," arXiv:1907.11692 2019.

[24] P. Liu, W. Yuan, J. Fu, Z. Jiang, H. Hayashi, G. Neubug. "Pre-train, Prompt, and Predict: A Systematic Survey of Prompting Methods in Natural Language Processing," *ACM Computing Surveys*, 55(9): Article 195, 2023.

[25] J. Lockerbie N. Maiden, J. Engmann, D. Randall, S. Jones, D. Bush, "Exploring the impact of software requirements on system-wide goals: a method using satisfaction arguments and i* goal modelling," *Requirements Engineering Journal*, 7:227–254, 2012.

[26] Y. Lu, M. Bartolo, A. Moore, S. Riedel, P. Stenetorp, "Fantastically Ordered Prompts and Where to Find Them: Overcoming Few-Shot Prompt Order Sensitivity," 60^{th} *Annual Meeting of the Association for Computational Linguistics*, pp. 8086–8098, 2022.

[27] N. Maiden, J. Lockerbie, D. Randall , S. Jones, D. Bush, "Using Satisfaction Arguments to Enhance i* Modelling of an Air Traffic Management System," *IEEE International Requirements Engineering Conference*, pp. 49-52, 2007.

[28] B. MacCartney, C.D. Manning. "Modeling Semantic Containment and Exclusion in Natural Language Inference," 22^{nd} *International Conference on Computational Linguistics (Coling)*, pp. 521–528, 2008.

[29] A. Murugesan, M.W. Whalen, E. Ghassabani, M.P.E. Heimdah, "Complete Traceability for Requirements in Satisfaction Arguments," *IEEE International Requirements Engineering Conference*, RE@Next!,, pp. 359-364, 2016.

[30] N. Nangia, C. Vania, R. Bhalerao, S.R. Bowman "CrowS-Pairs: A Challenge Dataset for Measuring Social Biases in Masked Language Models" *Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1953–1967, 2020.

[31] L. Ouyang, J. Wu, X. Jiang, et al. "Training language models to follow instructions with human feedback," *Advances in Neural Information Processing Systems (NeurIPS)*, 35, pp. 27730-44, 2022.

[32] J.W. Rae, S. Borgeaud, T. Cai, et al."Scaling Language Models: Methods, Analysis and Insights from Training Gopher," arXiv:2112.11446

[33] K. Ryan, "The Role of Natural Language in Requirements Engineering," *First IEEE International Symposium on Requirements Engineering*, pp. 240-242, 1993.

[34] T. Schick, J. Dwivedi-Yu, R. Dessì, R. Raileanu, M. Lomeli, R. Hambro, H. Zettlemoyer, N. Cancedda, T. Scialom. "Toolformer: Language Models Can Teach Themselves to Use Tools," *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

[35] W.R. Shadish, T.D. Cook, and D.T. Campbell. *Experimental and Quasi-experimental Designs for Generalized Causal Inference.* Houghton-Mifflin Company, Boston, Massachusetts, 2002.

[36] T. Sorensen, J. Robinson, C. Rytting, A. Shaw, K. Rogers, A. Delorey, M. Khalil, N. Fulda, D. Wingate. "An Information-theoretic Approach to Prompt Engineering Without Ground Truth Labels." *Association for Computational Linguistics (ACL)*, pp. 819–862, 2022.

[37] S.E. Toulmin, The Uses of Argument. Cambridge Univ. Press, 1958

[38] H. Trivedi, N. Balasubramanian, T. Khot, A. Sabharwal. "Interleaving Retrieval with Chain-of-Thought Reasoning for Knowledge-Intensive Multi-Step Questions," *Association for Computational Linguistics (ACL)*, pp. 10014–10037, 2023.

[39] A. van Lamsweerde. "Requirements engineering: from craft to discipline." 16^{th} *ACM SIGSOFT International Symposium on Foundations of software engineering (FSE)*, pp. 238–249, 2008.

[40] S. Wang, H. Fang, M. Khabsa, H. Mao, H. Ma. "Entailment as Few-Shot Learner," *Association for Computational Linguistics (ACL)*, pp. 13803–13817, 2023.

[41] X. Wang, J. Wei, D. Schuurmans, Q. V Le, E. H. Chi, S. Narang, A. Chowdhery, D. Zhou. "Self-Consistency Improves Chain of Thought Reasoning in Language Models," *International Conference on Learning Representations (ICLR),* 2023.

[42] L. Wang, W. Xu, Y. Lan, Z. Hu, Y. Lan, R. Ka-Wei Lee, E-P. Lim. "Plan-and-Solve Prompting: Improving Zero-Shot Chain-of-Thought Reasoning by Large Language Models," Association for Computational Linguistics (ACL), 2023.

[43] J. Wei, M. Bosma, V.Y. Zhao, K. Guu, A.W. Yu, B. Lester, N. Du, A.M. Dai, Q.V. Le, "Finetuned Language Models Are Zero-Shot Learners," *International Conference on Learning Representations*, 2022

[44] J. Wei, X. Wang, D. Schuurmans, M. Bosma, B. Ichter, F. Xia, E. Chi, Q. Le, D. Zhou. "Chain-of-Thought Prompting Elicits Reasoning in Large Language Models," *Advances in Neural Information Processing Systems* 35, pp. 24824-24837, 2022.

[45] J. Wei, D. Huang, Y. Lu, D. Zhou, Q.V. Le “Simple Synthetic Data Reduces Sycophancy in Large Language Models,” arXiv: 2308.03958 , 2023

[46] A. Williams, N. Nangia, S. Bowman. “A Broad-Coverage Challenge Corpus for Sentence Understanding through Inference,” *North American Chapter of the Association for Computational Linguistics (NAACL)*, 2018.

[47] S. Yao, D. Yu, J. Zhao, I. Shafran, T. L. Griffiths, Y. Cao, K.R. Narasimhan, “Tree of Thoughts: Deliberate Problem Solving with Large Language Models,” *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

[48] R.K. Yin. *Case study research*, 3rd ed. In Applied Social Research Methods Series, v.5. Sage Publications, 2003.

[49] P. Zave, M. Jackson. “Four Dark Corners of Requirements Engineering,” *ACM Transactions on Software Engineering and Methodology*, 6(1): 1-30, 1997.

[50] T.Z. Zhao, E. Wallace, S. Feng, D. Klein, S. Singh, “Calibrate Before Use: Improving Few-Shot Performance of Language Models,” 38^th *International Conference on Machine Learning* (PMLR) 139, 2021.

[51] D. Zhou, N. Schärli, L. Hou, J. Wei, N. Scales, X. Wang, D. Schuurmans, C. Cui, O. Bousquet, Q. Le, E. Chi, “Least-to-Most Prompting Enables Complex Reasoning in Large Language Models,” *International Conference on Learning Representations*, 2023