# Towards Detecting Inconsistencies in End-to-end Generated Task-Oriented Dialogues: A Constraint Satisfaction Approach

**Anonymous ACL submission**

## Abstract

Generative AI is profoundly transforming the core technologies behind conversational systems, shifting from component-based to end-to-end approaches. However, Large Language Models (LLMs) may still generate inconsistencies, a critical issue particularly in Task-Oriented Dialogues (TODs), where system responses must strictly adhere to information from a domain knowledge base (e.g., restaurants in a city). A single hallucination (e.g., suggesting a non-existent restaurant) can lead to severe task failures. We investigate a method for automatically detecting inconsistencies by conceptualizing TODs as a Constraint Satisfaction Problem (CSP), where variables represent dialogue segments referencing the conversational domain, and constraints among variables capture dialogue properties such as turn coherence and adherence to domain knowledge. We propose a pipeline that first identifies variables in a target dialogue and then applies a CSP solver to identify valid solutions. By comparing the target dialogue with valid variable assignments, we can detect inconsistencies and suggest minimal changes to ensure dialogue consistency. We demonstrate the high accuracy of the CSP-based approach in detecting inconsistencies, and provide a detailed analysis of our findings.

**Knowledge Base**

| ID | Name | Area | Food | Price |
|---|---|---|---|---|
| R1 | Taberna | centre | spanish | cheap |
| R2 | Espana | centre | spanish | moderate |
| R3 | Beirut | centre | lebanese | cheap |

**Dialogue**

| | |
|---|---|
| **User:** | I am looking for a restaurant serving **Spanish** food. |
| **System:** | There are three restaurants serving **Spanish** food, one is **cheap** and the other is **moderate** price range. Which price range would you prefer? |
| **User:** | I am looking for a **cheap** restaurant in **any** area that serves **Spanish** food. |
| **System:** | **Beirut** is **cheap** and serves Lebanese food. Would you like the location information? |

Figure 1: An inconsistent task-oriented dialogue (TOD) and a restaurant KB with slot-value pairs (N = Name, A = Area, F = Food, P = Price). Bold text highlights slot values in the dialogue. Red values indicate dialogic inconsistencies, while purple values indicate inconsistencies with the KB.

## 1 Introduction

Task-oriented dialogue (TOD) systems (McTear, 2020; Budzianowski et al., 2018; Balaraman et al., 2021; Qin et al., 2023) play a crucial role in human-computer interaction, facilitating seamless communication between users and machines to perform specific tasks. In recent years, transformer-based neural models have become the core technology behind TODs. In particular, pre-trained large language models (LLMs) allow end-to-end approaches ((Bang et al., 2023; Lai et al., 2023; Qin et al., 2023)) that greatly simplify the development of conversational systems, with respect to more complex component-based pipelines ((Young et al., 2013)). However, despite their impressive generative capabilities, it is well known that LLMs exhibit significant limitations in producing outputs that adhere to the requirements of task-specific domains ((Ji et al., 2022; Cho et al., 2022)). In a recent study, (Labruna et al., 2024), it has been shown that, when asked to generate a dialogue according to a given knowledge base ($KB$), as required by TODs, state-of-the-art open source LLMs produce

up to 59% of per dialogue disalignments with respect to the underlying KB. Failing to align their outputs with a domain $KB$, leads to inconsistencies that undermine LLMs reliability in real-world applications.

Figure 1 shows an example of a fragment of a Knowledge Base (three restaurants in a city) and a short TOD dialogue generated by a LLM. There are two hallucinations in this dialogue: first, at turn S1, the system mentions three restaurants serving Spanish food, which is not consistent with the knowledge base, where there are two such restaurants (this is a *domain inconsistency*). Second, at turn S2, the system introduces a Lebanese restaurant, which, although existing in the $KB$, it is not coherent with the previous dialogue turns, as a Spanish restaurant would have been expected (this is a *dialogic inconsistency*). Intuitively, both domain and dialogic inconsistencies need the whole dialogue context in order to be detected: for instance, *Lebanese* appears inconsistent because the user is looking for a Spanish restaurant since the beginning of the conversation, while considering turn S2 alone would result in a well formed dialogue. In addition, notice that three changes would make the whole dialogue consistent: (i) changing *three* with *two* at turn S1; (ii) changing *Lebanese* with *Spanish* at turn S2; and (iii) changing *Beirut* with *Taberna* at turn S2. Detecting TOD inconsistencies and, if possible, suggesting how to solve them, is the goal of this paper. The novel intuition of the paper is to consider dialogue consistency as a kind of *Constraint Satisfaction Problem* (*CSP* (Brailsford et al., 1999)), under the following working hypothesis: (i) first, dialogue consistency can be modeled with a limited number of domain independent constraints that need to be respected by appropriate linguistic realizations; (ii) such constraints can be well represented to define a CSP, whose allowed solutions can be identified by a CSP solver; (iii) a TOD is consistent if its linguistic realizations belong to the set of solutions allowed by a CSP solver for that dialogue. In the paper, we discuss how dialogue constraints are defined, how they can be extracted and modeled as a CSP, and how to set up an experimental setting where we can empirically prove that a CSP solver can detect inconsistencies in a dialogue and suggest possible changes that make the dialogue consistent.

The contributions of the paper are the following:

- We model TOD consistency as a Constraint Satisfaction Problem (CSP): to the best of our knowledge, this is a fully original approach.
- We set up a reusable experimental setting where TOD consistency can be automatically evaluated against a CSP solver.
- We show that the proposed CSP approach allows for effective detection of inconsistent TODs, achieving an accuracy of 75.9%.

## 2 Dialogue Consistency as a Constraint Satisfaction Problem

In this section, we explore the conceptualization of dialogue consistency in the CSP framework. We first describe the fundamental component of a conversational domain (Section 2.1), then we elucidate the various constraints that contribute to dialogue coherence (Section 2.2), encompassing linguistic, dialogic, and domain-based considerations. We finally expound upon the formalization of dialogue constraints as CSPs (Section 2.3), delineating the process of modeling dialogue coherence as a constraint satisfaction task.

### 2.1 TOD Conversational Domain

TODs typically need specific knowledge about the conversational domain (e.g., a database of restaurants, a playlist of songs, etc.). As in literature (Henderson et al., 2014), we assume a domain ontology providing a schema of the concepts (e.g., RESTAURANT, HOTEL, MOVIE), a set of slots $S$ (e.g., FOOD, AREA, PRICE) for the concepts, and the set of values that each slot can assume (e.g., EXPENSIVE, MODERATE, and CHEAP for the PRICE slot). Then, a domain knowledge base ($KB$) comprises a collection of instances for the ontology concepts, each consisting of $[slot - value]$ pairs, adhering to the domain ontology schema.

### 2.2 Dialogue Consistency

A TOD can be considered as a sequence of conversational turns between a user and a system aimed at achieving a specific goal. Within this framework, ensuring the consistency of the dialogue is crucial for effective communication between the user and the system. We consider three types of constraints, which need to be respected for a dialogue to be consistent: linguistic, dialogic and domain constraints.

**Linguistic Constraints.** They are necessary to respect general rules of language, including morpho-syntactic rules (e.g., genre and number agreement)

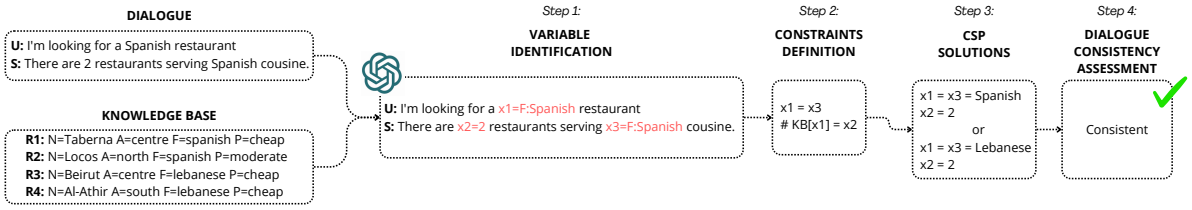


Figure 2: Overview of the CSP-based methodology. In step 1 GPT-4o is used to annotate the given dialogue for variable identification. Step 2 allocates the constraints that need to be true for the dialogue to be consistent. Step 3 uses CSP to find the possible solutions, and in step 4 the original dialogue is matched with the CSP solutions to assess its correctness.

and syntax-based rules (e.g., the correct use of a preposition). For instance, consider the following masked utterance:

> **U** I look for a restaurant in **<MASK>**.

The choice of *centre* as a substitute for the masked token is valid, whereas *expensive* would not be suitable because the preposition *in* is rarely used to introduce a price in English.

**Dialogic Constraints.** They maintain the semantic coherence across successive turns of the dialogue, ensuring that each utterance logically aligns with the preceding context, thereby facilitating a seamless flow of information. As an example, suppose the following masked dialogue turns:

> **U** I would like an Italian restaurant.
>
> **S** There is no **<MASK>** restaurant in the centre.

Here both *Italian* and *cheap* would be eligible choices from a linguistic point of view, but only *Italian* would maintain the coherence with the previous turn in the dialogue.

**Domain Constraints.** They ensure alignment between the dialogue content and the knowledge base of the system, thereby maintaining the dialogue's alignment with relevant factual information. Consider, for instance, a $KB$ with the following restaurants:

| ID | Name | Area | Food | Price |
|---|---|---|---|---|
| R1 | Mario | east | italian | expens. |
| R2 | Napoli | centre | italian | cheap |

And the following piece of masked dialogue:

> **U** I am looking for an **Italian** restaurant in the **centre**.
>
> **S** We have **<MASK>** restaurants available for your preferences.

Then, the only admissible choice for the masked token would be *one*, as selecting any other number would introduce an inconsistency with the information provided in the $KB$.

### 2.3 TOD Consistency as CSP

A Constraint Satisfaction Problem (CSP) (Brailsford et al., 1999; Kumar, 1992) imposes certain conditions on a finite set of variables through constraints. Each variable has a finite set of possible values, known as its domain, and constraints define the combinations of values allowed for specific subsets of the variables. A constraint can be given either explicitly, by enumerating the tuples allowed, or implicitly, e.g., by an algebraic expression. The solution of a CSP is an instantiation of all the variables for which all the constraints are satisfied. A CSP is solvable if it has at least one solution, otherwise it is unsolvable or overconstrained.

The hypothesis of this paper is that the dialogue constraints outlined in Section 2.2 can be modeled as CSPs. Intuitively, variables are the portions of the dialogue that need to be constrained (i.e., the $MASK$ tokens in our examples), while the range of possible values for the variables are expressed, either explicitly or implicitly, in the domain $KB$ for that dialogue. The CSP task consists of selecting variable assignments that comply with linguistic, dialogic, and domain constraints. To formalize this notion, consider a dialogue $d_i$ for which $n$ variables (i.e., masked tokens) $x_1, x_2, \ldots, x_n$ have been defined. Let $D_i$ denote the domain of possible values for variable $x_i$; let $\mathcal{C}$ be the set of constraints (i.e., linguistic,

dialogic, and domain constraints) over the dialogue $d_i$, and let $c$ represent a single constraint in $\mathcal{C}$. The CSP task is to determine if there exists an assignment $A = \{(x_1, a_1), (x_2, a_2), \ldots, (x_n, a_n)\}$ with $a_i \in D_i$ for $1 \leq i \leq n$, such that $A$ satisfies all constraints in $\mathcal{C}$. This problem can be formulated as follows:

$$\textit{Satisfies}(\{(x_1, a_1), (x_2, a_2), \ldots, (x_n, a_n)\}, C_j)$$

$$\forall C_j \in \mathcal{C}$$

where $\textit{Satisfies}(A, C_j)$ denotes the binary relationship between an assignment $A$ and a constraint $C_j$, indicating whether the assignment satisfies the constraint.

## 3 Methodology

This section outlines the process of modeling a TOD as a CSP, and then to assess the dialogue consistency using a CSP solver. The assessment involves three key steps for a $[d, KB]$ pair, where $d$ is a dialogue and $KB$ is a Knowledge Base: (1) identification of the variables within the dialogue $d$ (Section 3.1); (2) definition of dialogue constraints and construction of a CSP solver for the $[d, KB]$ pair (Section 3.2); and (3) application of the CSP solver to determine if the dialogue $d$ represents a feasible solution with respect to the defined constraints (Section 3.3). These phases of the methodology are illustrated in Figure 2.

### 3.1 Identifying TOD Variables

At step 1 (see Figure 2), we consider a TOD $d$ and a $KB$ (i.e., a set of entities described by slot-value pairs) related to the conversational domain of the dialogue. We do not assume any particular dependency between $d$ and $KB$: $d$ could be either fully covered by $KB$ (i.e., all mentions of slot values in $d$ are present in $KB$), only partially covered, or not covered at all. We consider CSP variables all text portions in $d$ either referring to a slot value in $KB$ or mentioning amounts of instances in $KB$. The rationale is that both slot values and instance amounts are elements that better characterize a TOD and are responsible for its consistency. In our example in Figure 1, we will obtain the following variables with their assignments, corresponding to highlighted tokens:

$[x_1 = Spanish]$, $[x_2 = three]$, $[x_3 = Spanish]$, $[x_4 = cheap]$ ... $[x_{11} = Lebanese]$.

### 3.2 Defining TOD Constraints

We have established a set $\mathcal{X}$ of variables $x_1, x_2, ..., x_n$, where each variable $x_i$ can assume a value either from the slot values or from amounts of instances in $KB$. Moving to step 2 in Figure 2, we now define the set of constraints $\mathcal{C}$ over the values that can be assigned to $\mathcal{X}$ variables. We consider the three categories of constraints introduced in Section 2.2: linguistic, dialogic, and domain-based constraints, and for each category we define a set of domain independent patterns, which are then instantiated as actual constraints on a TOD.

**Patterns for linguistic constraints.** We model linguistic constraints as the need for a variable derived from a slot value to match the semantic type of its slot type. For instance, given the utterance *I am looking for a restaurant at* $x_1$, the value of the variable $x_1$ must belong to the AREA type. More precisely, $C1$ is defined as follows:

$$C1 : x_1 \in V$$

where $V$ is the set of values belonging to the same slot type as the original value. Constraint $C1$, is meant to avoid that a variable can assume values that are semantically non valid. For instance, avoiding that $x_1$=NORTH can be assigned to a FOOD, as in *I am looking for a restaurant at* INDIAN, which is ungrammatical in English.

**Patterns for dialogic constraints.** We consider two dialogic constraints. $C2$ ensures that variables referring to the same slot-name and slot-value in $d$ are assigned to the same value. $C3$ ensures that variables with the same semantic type (i.e., same slot-name) occurring in the same utterance are assigned to different values. Given the turn *U: I want an* $x_1$ *restaurant. S: There are 3 restaurant that serve* $x_2$, we define $C2$ as follows:

$$C_2 : x_1 = x_2$$

where the aim is to keep internal coherence across the dialogue turns. Given the utterance *We have* $x_1$, $x_2$, *or* $x_3$ *restaurants.*, we define $C3$ as:

$$C_3 : x_1 \neq x_2, \quad x_1 \neq x_3, \quad x_2 \neq x_3$$

which captures non redundancy at the utterance level.

**Patterns for domain-based constraints.** We consider three domain-based constraints. All of them are meant to guarantee consistency between the number of instances mentioned in $d$ and the actual number of instances present in $KB$. We distinguish three cases: $C4$ covers the cases when an utterance in $d$ states that there are no instances in $KB$; $C5$ covers the cases where it is stated that there is at least one instance; and $C6$ the cases where there are exactly $n$ instances.

As for $C4$, consider an utterance indicating no results for a search: *There are no restaurants serving $x_1$ food*, assuming that there are no restaurants with [FOOD=$x_1$] in $KB$. For this utterance, $C4$ is defined as:

$$C4 : \neg\exists i \in KB \text{ with values } x_1$$

implying that the variable $x_1$ can not assume a value that is present in an instance of the $KB$.

As for $C5$, consider the utterance: *We have many $x_1$ restaurants at $x_2$*, where at least one restaurant with [FOOD=$x_1$] and [AREA=$x_2$] is supposed to exist in $KB$. For this utterance, $C5$ is defined as:

$$C5 : \exists i \in KB \text{ with values } x_1, x_2$$

imposing the existence of at least one instance with values $x_1$ and $x_2$. Finally, for $C6$, consider the utterance *There are $x_1$ restaurants at $x_2$*. We define the constraint as:

$$C6 : |\{i \in KB \text{ with value } x_2\}| = x_1$$

to check that the number of instances with value $x_2$ is exactly equal to $x_1$.

To sum up, we have defined six general, domain independent (i.e., in principle they can be applied to any TOD), constraint patterns over the variable of a TOD.

### 3.3 Assessing Dialogue Consistency

After identifying all variables and constraints for a dialogue $d$, a CSP solver computes all possible solutions for the variables in $d$ based on the knowledge base ($KB$) (step 3 in Figure 2). If one of these solutions matches the variable assignments in $d$, the dialogue is consistent with $KB$ (step 4 in Figure 2). For example, in Figure 1, the assignment $[x_2 = \text{three}]$ violates $C6$ (incorrect count of Spanish instances in $KB$), while $[x_{11} = \text{Lebanese}]$ violates $C2$ (lack of coherence with prior turns). If the CSP solver finds at least one solution, the variable assignments in the dialogue must match one of those solutions to ensure all constraints are satisfied. Conversely, if no solution exists with respect to $KB$, the variable assignments should either remain empty or include values not present in $KB$ to maintain consistency. When at least one solution is found but none matches the variable assignments, the solver identifies the most similar solution and determines the minimal changes required to make the dialogue consistent. This process provides a detailed report highlighting specific inconsistencies and suggesting corrections.

## 4 Validating CSP Performance

In this section we describe a controlled experiment aiming at assessing the ability of the CSP-based approach to detect inconsistencies in TODs, as depicted in Figure 2. We first create a balanced dataset of dialogues and corresponding $KB$s, such that half of the dialogues are inconsistent, and the other half is consistent. All [$d$, $KB$] pairs are then passed to a pipeline of components implementing the procedure in Figure 2. At the end of the process, the CSP component takes a binary decision about a [$d$, $KB$] pair: either it is CONSISTENT or NOT-CONSISTENT. The outcome is finally compared with the ground-truth for that pair for final assessment.

### 4.1 Dialogues and KB

As for [$d$, $KB$] pairs, we leverage MultiWoz 2.3 (MWoZ) (Han et al., 2020), a popular dataset for TODs. For our experiments we focus on 108 restaurant-related dialogues, from which 50% are randomly selected as the CONSISTENT ones (all MWoZ dialogues are by default consistent, as they were collected through human intervention)[1]. The remaining 50% of dialogues were randomly modified to make them inconsistent with respect to the MWoZ $KB$.

Then, for each dialogue $d$, a specific $KB$ is created selecting the pertinent restaurant instances from the global MWoZ $KB$, so that $KB$ is aligned with the content of $d$. Few instances of a $KB$ can be seen both in Figure 1 and in Appendix C.

### 4.2 CSP Variables and Constraints

Given a $[d, KB]$ pair, as described in the previous section, step 1 in Figure 2 involves variable

[1]Running the CSP approach, we discovered that 10% of MWoZ dialogues were actually inconsistent with the $KB$, because of human errors. Those dialogues were removed from the dataset.

identification.

As for constraints definition (step 2 in Figure 2) we instantiate the six constraint patterns introduced in Section 3.2 on the final set of variables identified by GPT-4o[2] in the dialogue $d$ and considering the $KB$ associate to $d$.

### 4.3 MiniZinc Constraint Solver

For the CSP solver (step 3 in Figure 2), we use MiniZinc (Nethercote et al., 2007), an open-source constraint programming language designed for modeling and solving constraint satisfaction problems. MiniZinc offers a high-level modeling language that enables users to define problem constraints and objectives, supporting a wide range of constraint types. This versatility makes it well-suited for diverse problem domains.

For our evaluation, we employed MiniZinc to obtain solutions that satisfy the dialogue constraints. Among its suite of solvers, we utilized Chuffed (Chu et al., 2018), a state-of-the-art solver renowned for its efficiency in solving CSPs. Chuffed's time-optimization capabilities are particularly advantageous for handling complex, large-scale optimization problems.

### 4.4 Methods and Evaluation Metrics

We employ four methods for classifying a [$d$, $KB$] pair either as a CONSISTENT or NOT-CONSISTENT. For all methods, results are calculated using accuracy, i.e., the proportion of correct guesses with respect to the total [$d$, $KB$] pairs to be classified.

**Random baseline.** By design, our dataset is balanced between CONSISTENT and NOT-CONSISTENT dialogues, therefore the random accuracy is 50%.

**MWoZ global variables + CSP.** In this method variables in $d$ are identified according to the semantic annotations provided by MWoZ and then passed to the CSP solver with the six constraints described in Section 5. If there is no valid assignment, then the dialogue is classified as NOT-CONSISTENT. This method, being based on the human annotations in MWoZ, is an upper-bound of the CSP performance.

**MWoZ local variables + CSP.** This baseline takes a decision considering variables in each turn in $d$ independently of the other turns. If at least one turn is inconsistent (i.e., the CSP solver does not find any valid assignment), then the whole dialogue is NOT-CONSISTENT. As in the previous method, variables are identified through MWoZ annotations.

**GPT4-o global variables + CSP.** This process uses GPT-4o, prompting the model to generate a JSON output closely aligned with the MWoZ annotation format. To ensure reliable variable extraction, we employ a *prompt chain* for each dialogue turn. The first prompt extracts candidate variables, while the second prompt verifies and refines them. More details on the prompts and the process are provided in Appendix B. The variables are identified at the global level, considering all turns in the dialogue. The intuition is that complex constraints need a context larger than a single turn.

| Method | accuracy (%) |
|---|---|
| RANDOM BASELINE | 50.0 |
| MWOZ GLOBAL VARIABLES + CSP | 91.6 |
| MWOZ LOCAL VARIABLES + CSP | 79.0 |
| GPT4-O GLOBAL VARIABLES + CSP | 75.9 |

Table 1: Results on assessing CSP performance.

### 4.5 The CSP-pipeline Effectiveness

Table 1 shows the results of the experiments. The upper-bound method based on MWoZ variables achieves 91.6 accuracy, correctly classifying 45 dialogues as CONSISTENT (83%) and 54 correctly as NOT-CONSISTENT (100%). At manual inspection, the 9 false negative dialogues show a misalignment between $d$ and the associated $KB$, due to human errors in labeling MWoZ. These results suggest that the CSP method is highly effective and provide a clear indication that the coverage of our six constraint template is very high. Notice that the same method on local variables drop performance to 79%, showing that TOD consistency require global approaches. Finally, the end-to-end CSP pipeline based on GPT-4o achieves 75.9% of accuracy. The drop in accuracy when using GPT-4o vs. MWoZ variable annotations is expected since GPT-4o is not perfect in the variable identification within the dialogues. As a result, this method produces 26 (25%) false negatives, highlighting an over application of constraints.

There are two main findings resulting from the presented experiments: (i) we implemented an end-to-end CSP-based pipeline able to achieve high accuracy in a task of dialogue inconsistency detection; (ii) the whole set of six constraint patterns

[2] https://openai.com/index/hello-gpt-4o/

| Dataset | # dialogues | # variables |
|---|---|---|
| ALL | 950 | 9281 |
| 1 SOLUTION | 18 | 54 |
| 2-10 SOLUTIONS | 134 | 868 |
| 11-100 SOLUTIONS | 286 | 2332 |
| 101+ SOLUTIONS | 306 | 3151 |

Table 2: Dialogue distribution based on CSP solutions (MiniZinc).

| Constraint | # variables | % coverage |
|---|---|---|
| C1 | **9281** | **100%** |
| C2 | 6084 | 66% |
| C3 | 1124 | 12% |
| C4 | 301 | 3% |
| C5 | 2369 | 26% |
| C6 | 4257 | 46% |

Table 3: Number and proportion of variables affected by each constraint.

defined in Section 3.2 has proven to be effective.

## 5 Analysing LLM Behavior

We analyse LLM behavior while generating dialogue inconsistencies. The main research questions are: (i) how do state-of-the-art LLMs behave when asked to generate consistent task-oriented dialogues (TODs)? (ii) How are the six constraint patterns involved in solving the CSP? (iii) Can we accurately localize the constraints responsible for TOD inconsistencies?

In the experiment, we use a corpus of 950 MultiWOZ dialogues across attractions, hotel, restaurant, and train domains. For each dialogue $d$, CSP variables are identified from MultiWOZ annotations, yielding a de-lexicalized dialogue $d_{delex}$. A LLM is then prompted to re-lexicalize $d_{delex}$ consistent with a knowledge base $KB$ associated with $d$, producing a re-lexicalized dialogue $d_{relex}$. Finally, the output is checked against the MiniZinc CSP solver (Section 4.3) to assess the model's capacity to satisfy our six constraints. This controlled setup investigates LLMs' ability to respect dialogue constraints.

### 5.1 Language Models

We evaluate four language models: LLaMA-3.1 8B (Dubey et al., 2024), GPT-3.5-Turbo (Achiam et al., 2023), GPT-4o (Hurst et al., 2024), and GPT-o1[3]. All models receive as input $d_{delex}$ and its associated $KB$ to generate consistent dialogues. Inference is zero-shot without fine-tuning, using the official APIs for closed-source models and HuggingFace checkpoints for open ones (details in Appendix A.1).

### 5.2 Baselines

We introduce four baselines for dialogue re-lexicalization (see Appendix A.2 for full details): RANDOM-ALL, RANDOM-SLOT, MOST FREQUENT-ALL, and MOST FREQUENT-SLOT. These baselines vary in how they assign slot values randomly or by frequency, either across all slots or restricted by slot type.

### 5.3 Evaluation Metrics

We use *Global Consistency Accuracy* (GCA) and *Variable Consistency Accuracy* (VCA) to evaluate adherence to constraints. GCA measures the proportion of dialogues fully satisfying all constraints, while VCA evaluates accuracy at the variable assignment level by comparing with the closest CSP solution (formal definitions in Appendix A.3).

Higher GCA and VCA indicate better dialogue consistency, beyond traditional metrics like BLEU or ROUGE. VCA can also localize specific errors by identifying slot-value assignments that cause violations.

### 5.4 Results

Table 2 shows the distribution of dialogues by the number of CSP solutions and the number of variables per group. Table 3 reports the impact of each constraint on the dialogue variables, showing $C1$ affects all variables, while $C4$ applies to fewer instances.

Table 4 compares baselines and LLM performance on re-lexicalizing TODs. GPT-4o and GPT-o1 outperform others significantly. Table 5 shows GPT-4o's performance by CSP solution group, with higher solution counts correlating with better consistency.

Table 6 (in Appendix A.4) shows an ablation study of constraints' impact on consistency metrics, highlighting the critical role of $C6$ (exact match with the number of $KB$ instances).

[3] https://openai.com/o1/

| Method | GCA | VCA |
|---|---|---|
| RANDOM-ALL | 0.01 | 0.02 |
| RANDOM-SLOT | 0.01 | 0.12 |
| MOST FREQUENT-ALL | 0.01 | 0.11 |
| MOST FREQUENT-SLOT | 0.06 | 0.23 |
| LLAMA-3.1 8B | 0.03 | 0.08 |
| GPT-3.5-TURBO | 0.11 | 0.37 |
| GPT-4O | **0.14** | 0.41 |
| GPT-O1 | **0.14** | **0.42** |

Table 4: Baselines and model performance on re-lexicalizing TODs.

| Split | GCA | VCA |
|---|---|---|
| 1 SOLUTION | 0.67 | 0.54 |
| 2-10 SOLUTIONS | 0.28 | 0.65 |
| 11-100 SOLUTIONS | 0.19 | 0.64 |
| 101+ SOLUTIONS | 0.11 | 0.64 |

Table 5: Re-lexicalizing TODs with GPT-4o across solution groups.

### 5.5 Discussion

The results highlight that GPT-4o and GPT-o1 outperform other models, though absolute GCA and VCA values remain moderate, indicating dialogue consistency remains challenging. Higher numbers of CSP solutions correlate with better consistency, suggesting that dialogues with multiple valid assignments are easier for LLMs to generate correctly. The ablation study confirms domain-based constraints as especially critical.

The detailed prompt format, baseline methodologies, extended metric definitions, and ablation study results are provided in Appendix A.

## 6 Related Work

TOD systems have been extensively investigated in NLP (Allen et al., 2001). Recent research has explored the use of neural network architectures for dialogue state tracking (Wu et al., 2020; Zhao et al., 2021) and policy learning (Su et al., 2016; Liu and Lane, 2017). Several metrics have been proposed to assess the performance of TOD systems, including task completion rates, user satisfaction scores, and objective measures for system components, such as precision, recall, and F1-score (Chen et al., 2017; Santhanam and Shaikh, 2019; Deriu et al., 2021). Recent studies have emphasized the importance of holistic evaluation frameworks that consider multiple aspects of dialogue quality (Zhang et al., 2021).

Maintaining consistency and coherence in dialogues is essential for effective communication between users and dialogue systems. Previous research has investigated various approaches to ensure dialogue coherence, including coherence modeling (Cervone et al., 2018), and coherence-based response generation (Cervone and Riccardi, 2020), aiming to enhance the naturalness and fluency of generated dialogues. Finally, several studies have explored the application of CSPs to language. These include early attempts to ensure coherence in generated text (Kibble and Power, 2004), model preposition lexicalization using constraints (Moriceau and Saint-Dizier, 2004), guide lexical choices through constraints (McKeown et al., 1997), and treat context-sensitive utterance generation as a CSP (Popescu et al., 2009). Differently to these works, our approach focuses on detecting inconsistencies in already generated TOD dialogues using CSP.

## 7 Conclusion

Generative LLMs may produce inconsistent TODs, due to misalignment between parametric memory and the TOD $KB$. We have introduced a novel approach to detect TOD inconsistencies based on Constraint Satisfaction. Several experiments demonstrate the feasibility of the approach, enabling to effectively identify and quantify inconsistencies present in TODs with high accuracy (75.9% with GPT-4o and CSP solver). We also analysed the LLM inconsistencies when tasked to re-lexicalize TODs, finding that they primarily concern domain knowledge adherence, resulting in an overall accuracy of only 0.14 at the dialogue level. Our study highlights the potential of CSP-based methodologies in evaluating dialogue consistency and identifying areas for improvement in automated dialogue generation systems. Future research should further explore the application of CSP in task-oriented dialogues and investigate strategies to enhance the coherence of LLM-generated dialogues, particularly in applications with strong domain knowledge requirements.

## Limitations

While the proposed Constraint Satisfaction Problem (CSP)-based approach offers a novel and effective method for detecting inconsistencies in task-

oriented dialogues (TODs), it presents several limitations.

The system relies on the explicit mapping of dialogues into variable-constraint representations. Although our method is domain-independent in principle, the process of extracting variables and constraints from dialogues may require customization or adaptation for new domains or dialogue schemas.

Our method focuses on identifying inconsistencies and suggesting minimal changes for correction, but it does not automatically regenerate fluent or user-aligned responses after such modifications. This leaves the generation of corrected natural language utterances as future work.

Finally, although our experimental results are promising, they are based on controlled datasets and manually designed inconsistencies. Further work is needed to assess robustness in more complex or organically generated dialogues.

## Ethical Considerations

**Use of Scientific Artifacts.** We used publicly available task-oriented dialogue datasets for experimentation. These datasets include MultiWOZ (Budzianowski et al., 2018) and variations based on it. Additionally, we used off-the-shelf large language models (LLMs) to generate new dialogues with intentional inconsistencies for controlled evaluation. All code developed for the CSP-based inconsistency detection pipeline is our original contribution and will be made publicly available for research purposes under an open-source license.

**Licensing and Intended Use.** All external datasets and models used in this work were employed in accordance with their licenses. Our use was consistent with the intended purpose of the datasets (research), and we explicitly specify that the CSP-based system and associated data artifacts are intended solely for research and educational use. Any derivative dataset created using our framework also inherits this research-only restriction.

**Privacy and Data Integrity.** The dialogue data used in this study does not include personally identifiable information (PII), and no effort was made to collect or infer such data. We manually verified the synthetic and benchmark dialogues for inappropriate or offensive content, and none was found. Our system does not involve any human annotation beyond the authors, so no consent or risk disclaimers were required.

**Documentation and Statistics.** All artifacts, including the experimental codebase, constraint templates, and synthetic dialogue generation scripts, are provided with the submission. This includes coverage across domains, types of slot-value inconsistencies, and linguistic patterns. We report the number of dialogue examples used in each experiment, as well as their train/test splits, in the experimental section. We also provide accuracy scores as descriptive statistics for the evaluation.

**Computational Resources.** Model generation and evaluation were conducted using a single NVIDIA A40 GPU, with a total budget of approximately 40 GPU hours. We do not fine-tune any large models; our work only uses them in inference mode.

**Use of Existing Software.** Our system uses standard NLP libraries such as Hugging Face Transformers and MiniZinc ‘constraint‘ solver library. All packages were used with default or explicitly documented parameters.

**Human Participants.** No human participants were recruited for this study, and no user studies or annotation tasks involving external contributors were conducted. Therefore, issues such as compensation or informed consent do not apply in our setting.

## References


Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, and 1 others. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

James Allen, George Ferguson, and Amanda Stent. 2001. An architecture for more realistic conversational systems. In *Proceedings of the 6th international conference on Intelligent user interfaces*, pages 1–8.

Vevake Balaraman, Seyedmostafa Sheikhalishahi, and Bernardo Magnini. 2021. Recent neural methods on dialogue state tracking for task-oriented dialogue systems: A survey. In *Proceedings of the 22nd Annual Meeting of the Special Interest Group on Discourse and Dialogue, SIGdial 2021, Singapore and Online, July 29-31, 2021*, pages 239–251. Association for Computational Linguistics.

Yejin Bang, Samuel Cahyawijaya, Nayeon Lee, Wenliang Dai, Dan Su, Bryan Wilie, Holy Lovenia, Ziwei Ji, Tiezheng Yu, Willy Chung, and 1 others. 2023. A multitask, multilingual, multimodal evaluation of chatgpt on reasoning, hallucination, and interactivity. *arXiv preprint arXiv:2302.04023*.

Sally C. Brailsford, Chris N. Potts, and Barbara M. Smith. 1999. Constraint satisfaction problems: Algorithms and applications. *European Journal of Operational Research*, 119(3):557–581.

Paweł Budzianowski, Tsung-Hsien Wen, Bo-Hsiang Tseng, Inigo Casanueva, Stefan Ultes, Osman Ramadan, and Milica Gašić. 2018. Multiwoz–a large-scale multi-domain wizard-of-oz dataset for task-oriented dialogue modelling. *arXiv preprint arXiv:1810.00278*.

Alessandra Cervone and Giuseppe Riccardi. 2020. Is this dialogue coherent? learning from dialogue acts and entities. *arXiv preprint arXiv:2006.10157*.

Alessandra Cervone, Evgeny Stepanov, and Giuseppe Riccardi. 2018. Coherence models for dialogue. *arXiv preprint arXiv:1806.08044*.

Hongshen Chen, Xiaorui Liu, Dawei Yin, and Jiliang Tang. 2017. A survey on dialogue systems: Recent advances and new frontiers. *Acm Sigkdd Explorations Newsletter*, 19(2):25–35.

Hyundong Cho, Chinnadhurai Sankar, Christopher Lin, Kaushik Ram Sadagopan, Shahin Shayandeh, Asli Celikyilmaz, Jonathan May, and Ahmad Beirami. 2022. Know thy strengths: Comprehensive dialogue state tracking diagnostics. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 5345–5359, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Geoffrey Chu, Peter J. Stuckey, Anthony Schutt, Thorsten Ehlers, Graeme Gange, and Keith Francis. 2018. Chuffed, a lazy clause generation solver. `https://github.com/chuffed/chuffed`.

Jan Deriu, Alvaro Rodrigo, Arantxa Otegi, Guillermo Echegoyen, Sophie Rosset, Eneko Agirre, and Mark Cieliebak. 2021. Survey on evaluation methods for dialogue systems. *Artificial Intelligence Review*, 54:755–810.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, and 1 others. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Ting Han, Ximing Liu, Ryuichi Takanobu, Yixin Lian, Chongxuan Huang, Wei Peng, and Minlie Huang. 2020. Multiwoz 2.3: A multi-domain task-oriented dataset enhanced with annotation corrections and co-reference annotation. *arXiv preprint arXiv:2010.05594*.

Matthew Henderson and 1 others. 2014. The second dialog state tracking challenge. In *Proceedings of the 15th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*, pages 263–272, Philadelphia, PA, U.S.A. Association for Computational Linguistics.

Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, and 1 others. 2024. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*.

Ziwei Ji and 1 others. 2022. Survey of hallucination in natural language generation. *ACM Computing Surveys*.

Rodger Kibble and Richard Power. 2004. Optimizing referential coherence in text generation. *Computational Linguistics*, 30(4):401–416.

Vipin Kumar. 1992. Algorithms for constraint-satisfaction problems: A survey. *AI magazine*, 13(1):32–32.

Tiziano Labruna, Sofia Brenna, Giovanni Bonetta, and Bernardo Magnini. 2024. Are you a good assistant? assessing llm trustability in task-oriented dialogues. *Clic-It 2024*.

Tuan M Lai, Giuseppe Castellucci, Saar Kuzi, Heng Ji, and Oleg Rokhlenko. 2023. External knowledge acquisition for end-to-end document-oriented dialog systems. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pages 3633–3647.

Bing Liu and Ian Lane. 2017. Iterative policy learning in end-to-end trainable task-oriented neural dialog models. In *2017 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pages 482–489. IEEE.

Kathleen McKeown, Michael Elhadad, and Jacques Robin. 1997. Floating constraints in lexical choice.

Michael McTear. 2020. Conversational ai: Dialogue systems, conversational agents, and chatbots. *Synthesis Lectures on Human Language Technologies*, 13(3):1–251.

Véronique Moriceau and Patrick Saint-Dizier. 2004. A constraint-based model for preposition choice in natural language generation. *Constraint Solving and Language Processing*, page 124.

Nicholas Nethercote, Peter J. Stuckey, Rowan Becket, Simon Brand, Greg J. Duck, and Guido Tack. 2007. Minizinc: Towards a standard cp modelling language. In *CP 2007*, volume 4741 of *LNCS*, pages 529–543. Springer.

Vladimir Popescu, Jean Caelen, and Corneliu Burileanu. 2009. A constraint satisfaction approach to context-sensitive utterance generation in multi-party dialogue systems. *International Journal of Speech Technology*, 12:95–112.

Libo Qin, Wenbo Pan, Qiguang Chen, Lizi Liao, Zhou Yu, Yue Zhang, Wanxiang Che, and Min Li. 2023. End-to-end task-oriented dialogue: A survey of tasks, methods, and future directions. In *Proceedings of the 2023 Conference on Empirical Methods in Natural*

*Language Processing*, pages 5925–5941, Singapore. Association for Computational Linguistics.

Sashank Santhanam and Samira Shaikh. 2019. Towards best experiment design for evaluating dialogue system output. *arXiv preprint arXiv:1909.10122*.

Pei-Hao Su, Milica Gasic, Nikola Mrksic, Lina Rojas-Barahona, Stefan Ultes, David Vandyke, Tsung-Hsien Wen, and Steve Young. 2016. On-line active reward learning for policy optimisation in spoken dialogue systems. *arXiv preprint arXiv:1605.07669*.

Peng Wu, Bowei Zou, Ridong Jiang, and AiTi Aw. 2020. Gcdst: A graph-based and copy-augmented multi-domain dialogue state tracking. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1063–1073.

Steve Young, Milica Gašić, Blaise Thomson, and Jason D Williams. 2013. Pomdp-based statistical spoken dialog systems: A review. *Proceedings of the IEEE*, 101(5):1160–1179.

Chen Zhang, Grandee Lee, Luis Fernando D'Haro, and Haizhou Li. 2021. D-score: Holistic dialogue evaluation without reference. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 29:2502–2516.

Jeffrey Zhao, Mahdis Mahdieh, Ye Zhang, Yuan Cao, and Yonghui Wu. 2021. Effective sequence-to-sequence dialogue state tracking. *arXiv preprint arXiv:2108.13990*.

# A Detailed Experimental Setup and Additional Results

## A.1 Prompting Details and Inference Setup

For dialogue re-lexicalization we employed four language models: LLaMA-3.1 8B, GPT-3.5-Turbo, GPT-4o, and GPT-o1. LLaMA-3.1 8B is a large-scale model fine-tuned for handling complex dialogue contexts and maintaining coherence in text generation (Dubey et al., 2024). GPT-3.5-Turbo is a model specifically designed for conversational tasks (Achiam et al., 2023). GPT-4o is an advanced language model recognized for its robust performance in various natural language processing tasks (Hurst et al., 2024). GPT-o1 is one of the latest update of the GPT series, designed to reason through complex tasks to solve harder problems[4]. All models were prompted with both the de-lexicalized dialogue, $d_{delex}$, and its associated $KB$ as input, ensuring a comprehensive context for producing dialogues that adhered to implicit constraints. Inference was conducted in zero-shot mode (see Appendix C) without fine-tuning, leveraging the respective APIs for closed source models and the huggingface checkpoints for the open ones: GPT-3.5-Turbo (2023-05-15), GPT-4o and GPT-o1 (2024-05-13), and LLaMA-3.1 8B (2023-07-10).

[4]https://openai.com/o1/

## A.2 Baselines

We introduce four dialogue re-lexicalization baselines:

- RANDOM-ALL generates a re-lexicalized dialogue $d_{relex}$ by randomly assigning variables in $d_{delex}$ to any slot values present in the $KB$, regardless of their slot type.
- RANDOM-SLOT also assigns variables randomly but restricts the selection to values associated with the same slot type as the original.
- MOST FREQUENT-ALL baseline assigns variables in $d_{delex}$ to the most frequent slot values found across all slots in the $KB$.
- MOST FREQUENT-SLOT baseline selects the most frequent value from the same slot type as the original.

## A.3 Evaluation Metrics

*Global Consistency Accuracy* (GCA) and *Variable Consistency Accuracy* (VCA) are the two metrics used to evaluate the adherence of a dialogue to a specific set of constraints. Given a re-lexicalized dialogue $d_{relex}$ where CSP variables are assigned to values, GCA measures the overall accuracy of the assignments for each variable. The average GCA is calculated as the proportion of dialogues that fully comply with all defined constraints:

$$GCA = \frac{\sum_{i=1}^{N} \left( \prod_{j=1}^{M} \textit{Satisfies}(A_i, C_j) \right)}{N}$$

where $N$ is the total number of dialogues, and $\textit{Satisfies}(A_i, C_j)$ is a binary indicator function that returns 1 if and only if all variable assignments in dialogue $d_i$ comply with the constraint $j$, 0 otherwise. On the other hand, VCA assesses the assignment accuracy on individual variables within the dialogue. We compare the dialogue assignment to the solutions of the CSP solver and find the most similar solution; then, we count how many variable assignments coincide with the assignments of the most similar solution. We formally define VCA as follows:

$$VCA = \frac{\sum_{i=1}^{N} |\textit{CorrectAssignments}(d_i)|}{M}$$

where $N$ is the total number of dialogues, $M$ is the total number of variables in the dialogues, and *CorrectAssignments*$(d_i)$ are the variable assignments in dialogue $d_i$ that coincide with the assignments of the most similar solution provided by the CSP solver. GCA and VCA provide insights into the ability of the dialogue generation system to maintain coherence and fidelity to the underlying domain knowledge while generating responses. Higher values of GCA and VCA indicate better performance in terms of dialogue quality and consistency, unlike traditional dialogue evaluation metrics (e.g., BLEU, ROUGE, or perplexity).

Additionally, the process used for computing VCA can be extended to identify specific errors within a dialogue. In cases where a dialogue is not among the solutions identified by the CSP, the most similar solution can be used to detect erroneous slot-value assignments. Specifically, errors are defined as slot-values that, if corrected, would result in a solution satisfying all constraints. This enables the generation of detailed reports pinpointing the errors in the dialogue, facilitating more targeted improvements.

### A.4 Ablation Study

Table 6 presents the results of an ablation study we conducted. The ablation study removes one constraint at a time to measure impact on GCA and VCA. Results indicate that $C6$ (exact match with KB instances) is the most critical, followed by $C1$ (hard constraints on slot values).

| Constraint | GCA | VCA |
|---|---|---|
| ALL EXCEPT C1 | 0.15 | 0.45 |
| ALL EXCEPT C2 | 0.15 | 0.42 |
| ALL EXCEPT C3 | 0.15 | 0.45 |
| ALL EXCEPT C4 | 0.15 | 0.46 |
| ALL EXCEPT C5 | 0.15 | 0.45 |
| ALL EXCEPT C6 | **0.21** | **0.48** |
| ALL EXCEPT DIALOGIC | 0.15 | 0.45 |
| ALL EXCEPT DOMAIN | **0.23** | **0.56** |

Table 6: Ablation study: global and variable consistency under different constraint configurations.

## B Variable Identification Prompt

The prompt chain used to annotate the dialogue turns consists of the following two prompts:

- **Prompt-1:** Analyze the given user utterance and extract any slot-value pairs. The possible slot types are: Area, Food, Price, Depart, Destination. Return the output as JSON with the dialog-act format.
- **Prompt-2:** Refine the given annotation for the user utterance. Ensure that only slots related to Area, Food, and Price are included. Correct any errors in the provided annotation, add missing slots, and remove any irrelevant slots. Return the output as JSON with the updated dialog-act format.

The GPT-4o response to Prompt-1 is used as input within Prompt-2, and the final output is a JSON file containing annotations about slot variables.

The *dialog-act* referred to in the two prompts is a JSON schema that guides GPT-4o in structured output mode and resembles the MultiWOZ JSON annotation schema.

## C Re-Lexicalization Prompt

Below is an instruction that outlines a task, along with a Knowledge Base containing domain-specific information to be utilized, and a dialogue for you to work on (see Figure 3).

**Instruction:**

Fill in the `[MASK]` placeholders in the dialogue based on the information provided in the Knowledge Base. Provide the updated dialogue exactly as it was given, but with the placeholders replaced by the appropriate values for each turn in the dialogue. If a turn does not contain any placeholders, leave the sentence unchanged. Turns should start with either *User* or *System*. Be aware of leaving blank spaces before punctuation as in the original (e.g., “Hi ,” instead of “Hi,”).

| ID | Name | Area | Food | Price |
|---|---|---|---|---|
| R1 | Mario | east | italian | expens. |
| R2 | Napoli | centre | italian | cheap |

**U** I'm looking for a restaurant serving **<MASK>** food in any area.

**S** There are no **<MASK>** restaurants in the area.

**U** Well, can I get the phone number to a **<MASK>** restaurant?

**S** Restaurant R2 serves **<MASK>** food. Their phone number is 01223 355166. Can I help you with anything else?

**U** That's it, goodbye.

**S** You're welcome, goodbye.

**U** I'm looking for a restaurant serving **italian** food in any area.

**S** There are no **italian** restaurants in the area.

**U** Well, can I get the phone number to a **cheap** restaurant?

**S** Restaurant R2 serves **italian** food. Their phone number is 01223 355166. Can I help you with anything else?

**U** That's it, goodbye.

**S** You're welcome, goodbye.

Figure 3: Example restaurants table and dialogue interactions.