Diagnosing Non-Executable Plans via Inconsistency Metrics: A KR-Inspired Perspective for Cognitive Robotics

Isabelle Kuhlmann[1] , **Mark O. Mints**[2] , **Peer Neubert**[2] , **Matthias Thimm**[1]
[1]University of Hagen, Germany
[2]University of Koblenz, Germany
isabelle.kuhlmann@fernuni-hagen.de, {mmints,neubert}@uni-koblenz.de,
matthias.thimm@fernuni-hagen.de

Abstract

Learning-based approaches to planning are increasingly used in robotics and cognitive systems. However, these methods do not guarantee the correctness or executability of their outputs. We introduce the notion of *inconsistency measurement* for planning, with a focus on quantifying the degree of inconsistency in non-executable plans. We formally define several measures that capture different aspects of plan-level inconsistency and illustrate their behavior through examples. Further, we discuss how the identification of inconsistencies can support the generation of human-understandable explanations. Our contributions provide a foundation for systematically analyzing inconsistency in planning systems.

1 Introduction

Recent advances in large language models (LLMs) have opened up new possibilities for robotics and cognitive systems, particularly in the area of task or action planning by introducing common-sense grounding and world context awareness (Song et al. 2023; Wang et al. 2024; Capitanelli and Mastrogiovanni 2024; Rajvanshi et al. 2024; Sathyamoorthy et al. 2024; Elnoor et al. 2024; Chu et al. 2025). The motivation behind this is that classical planning algorithms lack in the adaptability required for real-world applications, are generally computationally complex, and their application can quickly become too expensive in practical scenarios (Rana et al. 2023).

However, the use of learning-based models also introduces a new challenge: their outputs are not guaranteed to be correct. This is particularly critical in robotics, where an incorrect plan can result in non-executable behavior. At best, such a plan offers no practical utility; at worst, it may lead to failures or even unsafe situations during execution. To address this limitation, several recent works have explored hybrid approaches, in which LLM-generated plans are verified or refined using symbolic techniques from Knowledge Representation and Reasoning (KR) and human evaluation (Huang et al. 2022; Silver et al. 2022; Rana et al. 2023). While such methods are effective in filtering out inconsistent plans, they typically offer little insight into why a particular plan fails.

In this paper, we propose to address this gap through the lens of inconsistency measurement (Grant 1978; Grant and Martinez 2018; Thimm 2019), a subfield of KR focused on quantifying and analyzing logical conflicts. On one hand, a quantification of the degree of inconsistency in a given knowledge base (i. e., a set of logical formulas) allows human modelers to identify and compare alternative formalizations. On the other hand, such a quantification may also assist automated reasoning mechanisms. Furthermore, the analysis of the conflicts exhibited by a knowledge base may also help to later resolve the conflicts, in order to restore consistency.

Our central idea is to develop inconsistency measures for the domain of robotic planning as a tool for analyzing and explaining the inconsistency of (generated) plans. Note that this approach is not restricted to LLM-generated plans, but to any plan generated by a learning-based (e. g., (Asai and Fukunaga 2018; Aineto, Jiménez, and Onaindia 2018; Silver et al. 2021)) or approximate (e. g., (Baum, Nicholson, and Dix 2012; Heusner et al. 2014)) approach.

Our contributions can be summarized as follows:

- We introduce the concept of inconsistency measurement in (robotic) planning, including a discussion on different types of inconsistencies in this domain (Section 2).
- We propose 6 measures to quantify the degree of inconsistency in non-executable plans (Section 3).
- We demonstrate how our inconsistency measures can be leveraged to generate explanations in order to help human users to understand and correct existing conflicts (Section 4).

2 Background and Related Work

2.1 Inconsistency Measurement

In general, an *inconsistency measure* is simply a function that maps a knowledge base to a non-negative real value (or, in some cases, ∞). Most inconsistency measures in the literature are designed to measure inconsistency in propositional knowledge bases (i. e., sets of propositional logic formulas). Some measures are based on the notion of minimal inconsistent sets (Hunter and Konieczny 2008; Grant and Hunter 2011; Xiao and Ma 2012) or maximal consistent sets (Jabbour, Ma, and Raddaoui 2014; Ammoura et al. 2015), others are defined by making use of non-classical semantics (Knight 2002; Grant and Hunter 2011; Ma et al. 2009), and yet others view the models of the formulas in a knowledge base as points in Euclidian space (Grant

and Hunter 2013), or consider the minimal number of interpretations required to satisfy each formula in a given knowledge base (Thimm 2016). For a systematic overview, see the work by Thimm (2018).

However, inconsistency measurement has been applied to other target formalisms as well. Examples include the analysis of inconsistencies in news reports (Hunter 2006; Hunter and Summerton 2006), ontologies (Zhou et al. 2009; Zhang et al. 2017), and answer set programs (Ulbricht, Thimm, and Brewka 2016), the support of software requirements specifications (Martinez, Arias, and Vilas 2004; Zhu and Jin 2005), the handling of inconsistencies in business processes (Corea, Thimm, and Delfmann 2021; Corea, Grant, and Thimm 2022; Corea et al. 2024), inconsistency-tolerant reasoning in probabilistic logic (Potyka and Thimm 2017), and the monitoring and maintenance of quality in database settings (Decker and Misra 2017; Bertossi 2018; Parisi and Grant 2020; Parisi and Grant 2021; Parisi and Grant 2023; Grant and Parisi 2024).

In the context of planning, inconsistency measurement has received little attention—possibly because inconsistencies can arise at multiple distinct levels. We propose a preliminary taxonomy that distinguishes between the following types of inconsistency (see also Figure 1 for a visualization):

1. **Domain-level inconsistency:** The given domain specification itself is inconsistent. E. g., a predicate may simultaneously define a door as both open and closed, leading to logical contradictions or unreachable states.
2. **Task-level inconsistency:** The planning problem (task) is unsolvable, i. e., no valid plan exists that leads from the initial state to a goal state. E. g., a robot may be asked to retrieve water from a fridge, but the domain encodes that no water is available.
3. **Goal-level inconsistency:** A generated plan is executable but fails to achieve the desired goal condition. E. g., a robot is instructed to grab an apple, but the plan results in grabbing a banana instead.
4. **Execution-level inconsistency:** A generated plan contains one or more actions that are not applicable in the states in which they are invoked, i. e., the plan is not executable. E. g., a robot may be told to get water from a closed fridge, but the plan omits the action of opening the fridge first.

In this work, we focus on the latter level of inconsistency in planning. Note that inconsistency measurement on the level of domain specifications could be addressed by transforming the given constraints to a different logical formalism for which a number of inconsistency measures already exists in the literature. Moreover, there exists some work that is concerned with analyzing inconsistencies on the level of planning tasks. For instance, Eriksson, Röger, and Helmert (2018) provide a proof system for unsolvable planning tasks, while Sreedharan et al. (2019) and Sarwar, Ray, and Banerjee (2023) focus on explaining why a given task is unsolvable. Regarding the plan level, related work includes approaches to plan repair (Van Der Krogt and De Weerdt 2005; Fox et al. 2006) and fault diagnosis (Khalastchi and Kalech 2018; Das, Banerjee, and Chernova 2021), which aim to restore executability or explain failure. While these methods focus on correcting or explaining inconsistent plans, our work takes a complementary perspective rooted in KR by proposing formal inconsistency measures that quantify how inconsistent a non-executable plan is.

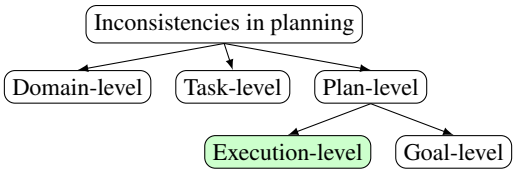

Figure 1: A taxonomy of inconsistencies in classical planning.

2.2 Classical Planning in Cognitive Robotics

To lay a clear foundation for the study of inconsistency in planning, we adopt a classical planning model (Ghallab, Nau, and Traverso 2004) in this work. While more expressive variants exist, starting with a simple setting allows us to systematically develop formal inconsistency measures. To be precise, our focus is on fully observable, deterministic environments with known initial and goal states, and we restrict ourselves to sequential plans without explicit modeling of time. Future work could extend this approach to richer models, such as temporal or partially observable planning systems, or multi-agent settings.

We consider a *restricted state transition system* $\Sigma = (S, A, \gamma)$, with S being a finite set of world states, A being a finite set of actions that can be performed, and γ being a function which deterministically maps a state s_1 to another state $s_2 = \gamma(s_1, a)$, with $s_1, s_2 \in S$, and $a \in A$. To concretely represent planning problems, this abstract structure is instantiated over a finite set of predicate symbols $L = \{p_1, \ldots, p_m\}$ and a finite set of objects $O = \{o_1, \ldots, o_k\}$, resulting in a (set-theoretic) *planning domain*. Thus, a planning domain on L is a state transition system $\Sigma = (S, A, \gamma, L, O)$ for which the following holds.

- The set of propositions used in states and actions is obtained by grounding the predicate symbols in L with the objects in O, resulting in the set $L_G = \{p(o_1, \ldots, o_k) \mid p \in L, p \text{ has arity } k, \text{ and } o_1, \ldots, o_k \in O\}$.
- Each state $s \in S$ is a subset of L_G, i. e., $S \subseteq 2^{L_G}$. Let $p \in L$ and $s \in S$. If $p \in s$ it means that p holds in s; likewise, if $p \notin s$ it means that p does not hold in s.
- An action $a \in A$ is determined by a set of preconditions $\mathrm{Pre}(a)$, a set of *add* effects $\mathrm{Eff}^+(a)$, and a set of *delete* effects $\mathrm{Eff}^-(a)$. Formally, a is defined as $a = (\mathrm{Pre}(a), \mathrm{Eff}^+(a), \mathrm{Eff}^-(a))$, with $\mathrm{Eff}^+(a) \cap \mathrm{Eff}^-(a) = \emptyset$. Further, a is *applicable* to a state s if $\mathrm{Pre}(a) \subseteq s$.
- Intuitively, w.r.t. a given action a and a state s, the state transition function γ calculates the next state by deleting $\mathrm{Eff}^-(a)$ from s and by adding $\mathrm{Eff}^+(a)$ to it. Thus, formally, $\gamma(s, a) = (s \setminus \mathrm{Eff}^-(a)) \cup \mathrm{Eff}^+(a)$, given that

$\mathrm{Pre}(a) \subseteq s$. If $\mathrm{Pre}(a) \not\subseteq s$ (i. e., a is not applicable to s), $\gamma(s, a)$ is undefined.

- If a is applicable to a state s, it produces another state, i. e., if $\mathrm{Pre}(a) \subseteq s$, then $(s \setminus \mathrm{Eff}^-(a)) \cup \mathrm{Eff}^+(a) \in S$.

A *planning problem* is denoted as $\mathcal{P} = (\Sigma, s_0, g)$, with $s_0 \in S$ being the *initial state*, and $g \subseteq L$ being a set of *goal propositions* that determine the set of goal states $S_g = \{s \in S \mid g \subseteq s\}$. In other words, g specifies what a state s must satisfy in order to be a goal state.

We define a *plan* to be any sequence of actions $\mathcal{A} = \langle a_0, \ldots, a_{n-1} \rangle$ with $n > 0$, and its *length* is $|\mathcal{A}| = n$. Moreover, we define a plan $\mathcal{A}$ to be a solution to a given planning problem $\mathcal{P} = (\Sigma, s_0, g)$ if $g \subseteq \gamma(s_0, \mathcal{A})$, i. e., $\mathcal{A}$ corresponds to a sequence of state transitions $s_0, s_1, \ldots, s_n$, such that $s_{i+1} = \gamma(s_i, a_i)$ (with $i \in \{0, \ldots, n-1\}$), and $s_n \in S_g$. Note that in classical planning, one often considers optimality in addition to executability. To this end, each action gets assigned a cost, and an optimal plan is one with a minimum total cost. However, in this work, we do not consider optimality and focus exclusively on executability.

In the context of cognitive robotics, the objective is often to produce a valid plan—that is, a correct solution to a given planning problem—in order to execute a task in the physical world. However, as noted in the introduction, learning-based approaches such as LLMs do not generally guarantee correctness, and may produce flawed or incomplete plans. To reason about such situations, we distinguish between *consistent* and *inconsistent* plans. As mentioned before in Section 2.1, in this work, we focus exclusively on the executability of plans, which we take as the primary dimension of consistency. Specifically, we define a plan as *consistent* (also referred to as *executable*) if every action in the sequence is applicable in the state where it is invoked, according to the transition model of the domain. A plan is *inconsistent* (also referred to as *non-executable*) if at least one action in the sequence has unsatisfied preconditions at the time of execution, i. e., the plan cannot be executed in full from the initial state. This definition intentionally abstracts away from goal achievement. While a plan that does not reach a goal state may still be considered incomplete or suboptimal, we do not regard it as inconsistent for the purposes of this paper. Our focus is on capturing and analyzing failures that arise from structural flaws in the plan itself, rather than from external goal misalignment.

To illustrate the definitions above, we introduce a simple example from the well-known *Blocks World* planning domain in Example 1. Note that we will use this domain throughout this paper as a running example. For further information on planning, see, e. g., (Ghallab, Nau, and Traverso 2004).

Example 1. *We consider the famous* Blocks World *planning domain (Slaney and Thiébaux 2001), which consists of a finite number of blocks on a table, that can be stacked on top of each other (by means of a mobile robot). In the initial state, some blocks might already be stacked, and the task is to reach a specific set of stacks of blocks. Only one block can be moved at a time, and a block can only be placed on the table or on another block.*

We specify a Blocks World planning problem as follows. Block A *is initially stacked on* B*,* B *is stacked on* C*, and* C *is on the table. Let the task be to have the following stack (from top to table):* B, A, C*; see the leftmost and rightmost parts of Figure 2 for an illustration. We define the set L_b of predicates as*

$$L_b = \{\texttt{on}(x, y), \texttt{onTable}(x), \texttt{clear}(x), \texttt{holding}(x), \texttt{gripperEmpty}\} \quad \textit{with}$$

- $\texttt{on}(x, y)$ *meaning that block x is on block y;*
- $\texttt{onTable}(x)$ *meaning that block x is on the table;*
- $\texttt{clear}(x)$ *meaning that there is no other block on block x, and block x can therefore be gripped;*
- $\texttt{holding}(x)$ *meaning that the robot is holding block x;*
- $\texttt{gripperEmpty}$ *meaning that the robot's gripper is empty, i. e., it is currently not holding a block.*

Moreover, we represent the blocks A, B, C *as the set of objects $O_b = \{A, B, C\}$, respectively. We now define the planning domain $\Sigma_b = (S_b, A_b, \gamma, L_b, O_b)$. In order to obtain A_b, we define the actions $\texttt{unstack}(x, y)$, which specifies how a block x is gripped from another block y, and $\texttt{stack}(x, y)$, which specifies how a block x is placed on another block y. Further, $\texttt{unstackT}(x)$ specifies that block x is picked up from the table, and $\texttt{stackT}(x)$ specifies that block x is placed on the table. Formally, the actions in A_b are defined as follows:*

$$\begin{aligned}
\texttt{unstack}(x, y) = (&\{\texttt{on}(x, y), \texttt{clear}(x), \texttt{gripperEmpty}\},\\
&\{\texttt{holding}(x), \texttt{clear}(y)\},\\
&\{\texttt{on}(x, y), \texttt{clear}(x), \texttt{gripperEmpty}\})\\
\texttt{stack}(x, y) = (&\{\texttt{holding}(x), \texttt{clear}(y)\},\\
&\{\texttt{on}(x, y), \texttt{clear}(x), \texttt{gripperEmpty}\},\\
&\{\texttt{holding}(x), \texttt{clear}(y)\})\\
\texttt{unstackT}(x) = (&\{\texttt{onTable}(x), \texttt{clear}(x), \texttt{gripperEmpty}\},\\
&\{\texttt{holding}(x)\},\\
&\{\texttt{onTable}(x), \texttt{clear}(x), \texttt{gripperEmpty}\})\\
\texttt{stackT}(x) = (&\{\texttt{holding}(x)\},\\
&\{\texttt{onTable}(x), \texttt{clear}(x), \texttt{gripperEmpty}\},\\
&\{\texttt{holding}(x)\})
\end{aligned}$$

We now define a planning problem $\mathcal{P}_1 = (\Sigma_b, s_0, g_1)$ with the initial state $s_0 = \{\texttt{onTable}(C), \texttt{on}(B, C), \texttt{on}(A, B), \texttt{clear}(A), \texttt{gripperEmpty}\}$ and $g_1 = \{\texttt{onTable}(C), \texttt{on}(A, C), \texttt{on}(B, A)\}$. Note that the only state $s \in S_b$ with $g_1 \subseteq s$ that is actually reachable w.r.t. A_b and γ (i. e., the state shown in the rightmost part of Figure 2) is $\{\texttt{onTable}(C), \texttt{on}(A, C), \texttt{on}(B, A), \texttt{clear}(B), \texttt{gripperEmpty}\} \supseteq g_1$.

We now examine plan $\mathcal{A}_1$, which is consistent and ends in a valid goal state, i. e., it is also a solution to $\mathcal{P}_1$ (see again Figure 2 for an illustration of each state and action):

$$\begin{aligned}
\mathcal{A}_1 = \langle &\texttt{unstack}(A, B), \texttt{stackT}(A), \texttt{unstack}(B, C),\\
&\texttt{stackT}(B), \texttt{unstackT}(A), \texttt{stack}(A, C),\\
&\texttt{unstackT}(B), \texttt{stack}(B, A)\rangle
\end{aligned}$$

Now consider plan $\mathcal{A}_2$, which is the same sequence of actions as $\mathcal{A}_1$, except that the second action ($\texttt{stackT}(A)$)

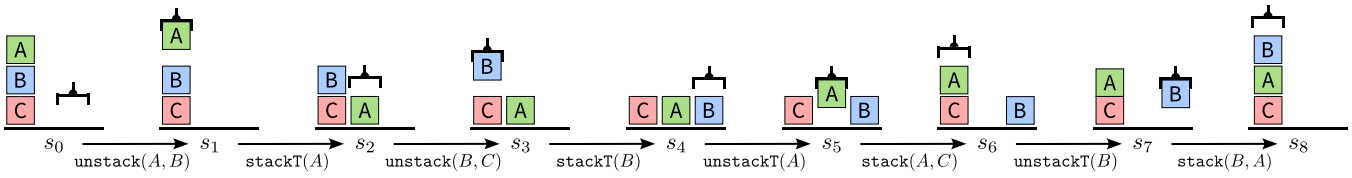

Figure 2: Blocks World plan $\mathcal{A}_1$ from Example 1.

is missing, i. e., $\mathcal{A}_2 = \langle \texttt{unstack}(A, B), \cancel{\texttt{stackT}(A)}, \texttt{unstack}(B, C), \ldots \rangle$. *According to this plan, we first apply* $a_0 = \texttt{unstack}(A, B)$ *to* s_0*, and we get:*

$$\begin{aligned} s_1 &= \gamma(s_0, a_0) = (s_0 \setminus \text{Eff}^-(a_0)) \cup \text{Eff}^+(a_0) \\ &= (s_0 \setminus \{\texttt{on}(A, B), \texttt{clear}(A), \texttt{gripperEmpty}\}) \\ &\quad \cup \{\texttt{holding}(A), \texttt{clear}(B)\} \\ &= \{\texttt{onTable}(C), \texttt{on}(B, C), \texttt{holding}(A), \texttt{clear}(B)\} \end{aligned}$$

Next, $a_1 = \texttt{unstack}(B, C)$ *needs to be applied to* s_1*; however, the preconditions of* a_1 *are not satisfied (i. e.,* a_1 *is not applicable w.r.t.* s_1*), since* $\text{Pre}(a_1) = \{\texttt{on}(B, C), \texttt{clear}(B), \texttt{gripperEmpty}\} \not\subseteq s_1$*. Thus, there exists an action in* $\mathcal{A}_2$ *that cannot be applied, and, consequently,* $\mathcal{A}_2$ *is inconsistent.*

3 Inconsistency Measures for Plans

We now turn to the central contribution of this work: the definition of inconsistency measures for non-executable plans. To focus specifically on execution-level issues, we impose the assumptions that the given planning domain is consistent, and the underlying planning problem is solvable; in addition, we do not consider whether the final state is actually a goal state. Let $\mathbb{P}$ be the set of all planning problems, let $\mathbb{A}$ be the set of all plans, and let $\mathbb{R}^{\infty}_{\geq 0} := \mathbb{R}_{\geq 0} \cup \{\infty\}$.

Definition 1. *A* plan inconsistency measure *is a function* $\mathcal{I} : \mathbb{P} \times \mathbb{A} \to \mathbb{R}^{\infty}_{\geq 0}$.

We further require that $\mathcal{I}(\mathcal{P}, \mathcal{A}) = 0$ if the given plan $\mathcal{A}$ is executable w.r.t. the given problem $\mathcal{P}$ (i. e., there is no inconsistency). Note that, in the remainder of this paper, we may refer to a *plan inconsistency measure* simply as an *inconsistency measure* or even just a *measure*.

Further, note that plan inconsistency values are technically allowed to be ∞. In propositional logic, not all measures include this value, however, in some definitions it is necessary as no (finite) value can be computed in some cases. We include ∞ in our definition in order for it to be in line with similar definitions from the literature (in particular w.r.t. propositional logic), and for it to allow for greater generality and extensibility. Future work may explore measures that, for instance, assign ∞ to (non-executable) plans for unsolvable planning problems.

Drastic Inconsistency Measure As a first step, we define a *drastic* plan inconsistency measure, which is in line with the drastic inconsistency measure for propositional logic (Hunter and Konieczny 2008). In the propositional case, this measure returns 1 if the given knowledge base is inconsistent and 0 otherwise (i. e., if it is consistent). Thus, analogously, a the drastic plan inconsistency measure returns 1 if the plan is not executable and 0 otherwise, i. e., if it is in fact executable and there is no inconsistency.

Definition 2. *The* drastic inconsistency measure $\mathcal{I}_{\text{d}}$ *is defined as*

$$\mathcal{I}_{\text{d}}(\mathcal{P}, \mathcal{A}) = \begin{cases} 1 & \text{if } \exists a_i \in \mathcal{A} : \text{Pre}(a_i) \not\subseteq s_i, \\ & i \in \{0, \ldots, |\mathcal{A}| - 1\} \\ 0 & \text{otherwise} \end{cases}$$

w.r.t. a given planning problem $\mathcal{P} = (\Sigma, s_0, g)$*, with* $\Sigma = (S, A, \gamma, L, O)$ *being a planning domain, a given plan* $\mathcal{A} = \langle a_0, \ldots, a_{n-1} \rangle$*, and* $s_{i+1} = \gamma(s_i, a_i)$*.*

Example 2. *Consider again the plans* $\mathcal{A}_1$ *and* $\mathcal{A}_2$ *from Example 1. With regard to* $\mathcal{A}_1$*, each action* a_i *is applicable to its corresponding state* s_i *(with* $i \in \{0, \ldots, |\mathcal{A}_1| - 1\}$*). Thus, it is executable and therefore consistent, and we get* $\mathcal{I}_{\text{d}}(\mathcal{P}_1, \mathcal{A}_1) = 0$*. With regard to* $\mathcal{A}_2$*, we already showed that* $\text{Pre}(a_1) \not\subseteq s_1$*, i. e.,* $\exists i : a_i \in \mathcal{A}_2$ *such that* $\text{Pre}(a_i) \not\subseteq s_i$*, and we get* $\mathcal{I}_{\text{d}}(\mathcal{P}_1, \mathcal{A}_2) = 1$*.*

First-Conflict Inconsistency Measures As the drastic inconsistency measure is not very expressive and is not very helpful in actually analyzing an inconsistency, we define a more informative measure next. More specifically, this measure is based on the idea that a plan in which an invalid action occurs early on could be considered "more inconsistent" than a plan in which the first invalid action occurs later. To quantify this idea, we search for the index of the first invalid action and subtract it from the total number of actions.

Definition 3. *The* first-conflict inconsistency measure $\mathcal{I}_1$ *is defined as*

$$\mathcal{I}_1(\mathcal{P}, \mathcal{A}) = \begin{cases} |\mathcal{A}| - i & \text{if } \exists a_i \in \mathcal{A} : \text{Pre}(a_i) \not\subseteq s_i \text{ and} \\ & \nexists a_j \in \mathcal{A} : \text{Pre}(a_j) \not\subseteq s_j \text{ and } j < i \\ & \text{with } i, j \in \{0, \ldots, |\mathcal{A}| - 1\} \\ 0 & \text{otherwise} \end{cases}$$

w.r.t. a given planning problem $\mathcal{P} = (\Sigma, s_0, g)$*, with* $\Sigma = (S, A, \gamma, L, O)$ *being a planning domain, a given plan* $\mathcal{A} = \langle a_0, \ldots, a_{n-1} \rangle$*, and* $s_{i+1} = \gamma(s_i, a_i)$*.*

Note that we defined this measure as an absolute measure, however, it could easily be adapted to become a relative measure by normalizing it, i. e., by dividing the resulting value by $|\mathcal{A}|$. For the sake of simplicity, we only consider absolute measures in this paper; nevertheless we refer the interested reader to the work by Besnard and Grant (2020) for a discussion on absolute vs. relative inconsistency measures.

Example 3. *Consider again $\mathcal{A}_2$ from Example 1. In addition, consider the plan $\mathcal{A}_3$, which is defined exactly as $\mathcal{A}_1$, except that the action* $\texttt{unstackT}(B)$ *(the second to last action) is omitted, i. e., $\mathcal{A}_3 = \langle \ldots, \texttt{stack}(A, C), \cancel{\texttt{unstackT}(B)}, \texttt{stack}(B, A)\rangle$. W.r.t. $\mathcal{A}_2$, action $a_1 = \texttt{unstack}(B, C)$ is not applicable, due to the lack of* $\texttt{stackT}(A)$ *beforehand. More precisely, we have $s_1 = \{\texttt{onTable}(C), \texttt{on}(B, C), \texttt{holding}(A), \texttt{clear}(B)\}$ and $\mathrm{Pre}(a_1) = \{\texttt{on}(B, C), \texttt{clear}(B), \texttt{gripperEmpty}\}$, and thus, $\texttt{gripperEmpty} \in \mathrm{Pre}(a_1)$, but $\texttt{gripperEmpty} \notin s_1$. W.r.t. $\mathcal{A}_3$, action $a_6 = \texttt{stack}(B, A)$ is not applicable, due to the lack of* $\texttt{unstackT}(B)$*. Thus, we get $\mathcal{I}_1(\mathcal{P}_1, \mathcal{A}_2) = |\mathcal{A}_2| - 1 = 7 - 1 = 6$ and $\mathcal{I}_1(\mathcal{P}_1, \mathcal{A}_3) = 7 - 6 = 1$.*

The previously introduced first-conflict measure only captures the index at which a plan first becomes non-executable. However, it does not take into account how many preconditions of the corresponding action are actually violated. As a result, two plans (of the same length) that both fail at the same index receive the same inconsistency value—even if the first plan violates only a single precondition, while the second violates, say, ten. To address this limitation, we propose to define a refined version of the first-conflict measure that additionally considers the number of preconditions that are not satisfied. One way to achieve this is by multiplying the value of $\mathcal{I}_1$ by the number of unsatisfied preconditions, as shown in the following definition.

Definition 4. *Let $\mathcal{A} = \langle a_0, \ldots, a_{n-1}\rangle$ be a given plan, and let $\varphi(\mathcal{A}, i) = (|\mathcal{A}| - i) \cdot |\mathrm{Pre}(a_i) \setminus s_i|$ with $i \in \{0, \ldots, |\mathcal{A}| - 1\}$. The* precondition-based *first-conflict inconsistency measure $\mathcal{I}_1^{\mathrm{p}}$ is defined as*

$$\mathcal{I}_1^{\mathrm{p}}(\mathcal{P}, \mathcal{A}) = \begin{cases} \varphi(\mathcal{A}, i) & \text{if } \exists a_i \in \mathcal{A} : \mathrm{Pre}(a_i) \not\subseteq s_i \text{ and} \\ & \nexists a_j \in \mathcal{A} : \mathrm{Pre}(a_j) \not\subseteq s_j \text{ and } j < i \\ & \text{with } i, j \in \{0, \ldots, |\mathcal{A}| - 1\} \\ 0 & \text{otherwise} \end{cases}$$

w.r.t. a given planning problem $\mathcal{P} = (\Sigma, s_0, g)$, with $\Sigma = (S, A, \gamma, L, O)$ being a planning domain, a given plan $\mathcal{A} = \langle a_0, \ldots, a_{n-1}\rangle$, and $s_{i+1} = \gamma(s_i, a_i)$.

Example 4. *Consider again $\mathcal{A}_2$ from Example 1. As explained in Example 3, action $a_1 = \texttt{unstack}(B, C)$ is not applicable, because $\texttt{gripperEmpty} \in \mathrm{Pre}(a_1)$, but $\texttt{gripperEmpty} \notin s_1$, i. e., $\mathrm{Pre}(a_1) \not\subseteq s_1$. As the other preconditions of a_1 are included in s_1, exactly one precondition is not satisfied, and we get $\mathcal{I}_1^{\mathrm{p}}(\mathcal{P}_1, \mathcal{A}_2) = (7 - 1) \cdot 1 = 6$. Now consider $\mathcal{A}_4$, which is defined exactly as $\mathcal{A}_2$, but a_1 is replaced by $a_1' = \texttt{unstack}(A, C)$. Thus, we have $\mathrm{Pre}(a_1') = \{\texttt{on}(A, C), \texttt{clear}(A), \texttt{gripperEmpty}\}$. As none of these preconditions is in s_1, we get $\mathcal{I}_1^{\mathrm{p}}(\mathcal{P}_1, \mathcal{A}_4) = (7 - 1) \cdot 3 = 18$.*

Note that other aggregation methods are also possible; for instance, using an additive formulation instead of a multiplicative one. The version introduced here is meant to illustrate the general idea of additionally incorporating the number of violated preconditions.

Error-Propagation Inconsistency Measures The previously defined first-conflict measures consider only the first point in the plan where an action is not executable. The motivation behind this is that, once there is an action that is not executable, the entire plan cannot be executed. However, when aiming to evaluate how severely a plan deviates from consistency, it can also be useful to look beyond the first-conflict and assess how errors would propagate throughout the remainder of the plan. In other words, we are interested in how a single conflict influences subsequent steps, and whether additional inconsistencies occur further down the line. Therefore, we now introduce an inconsistency measure that propagates errors through the entire plan and counts the total number of actions for which at least one precondition is not satisfied. To achieve this, we define a relaxed transition function $\tilde{\gamma}$ that does not enforce preconditions. Unlike the original transition function γ, which only applies an action if its preconditions are satisfied, $\tilde{\gamma}$ applies the effects of an action unconditionally. Hence, setting $s_{i+1} = \tilde{\gamma}(s_i, a_i)$ for $i \in \{0, \ldots, n - 1\}$ allows us to compute a sequence of (possibly inconsistent) states $s_0, \ldots, s_n$ that reflects the theoretical progression of the plan $\mathcal{A} = \langle a_0, \ldots a_{n-1}\rangle$ from the initial state s_0, regardless of whether it is executable.

Definition 5. *The* error-propagation inconsistency measure $\mathcal{I}_{\mathrm{e}}$ *is defined as*

$$\mathcal{I}_{\mathrm{e}}(\mathcal{P}, \mathcal{A}) = \sum_{i=0}^{|\mathcal{A}|-1} \begin{cases} 1 & \text{if } \mathrm{Pre}(a_i) \not\subseteq s_i \\ 0 & \text{otherwise} \end{cases}$$

w.r.t. a given planning problem $\mathcal{P} = (\Sigma, s_0, g)$, with $\Sigma = (S, A, \tilde{\gamma}, L, O)$ being a planning domain, a given plan $\mathcal{A} = \langle a_0, \ldots, a_{n-1}\rangle$, and $s_{i+1} = \tilde{\gamma}(s_i, a_i)$.

Example 5. *Consider again $\mathcal{A}_2$ from Example 1. As shown in Example 3, action a_1 is not executable, because one precondition is not satisfied. When propagating the plan further (using the relaxed transition function $\tilde{\gamma}$), we see that for a_2, all preconditions are satisfied. However, w.r.t. a_3, two preconditions are violated. From this point on, the resulting state s_4 corresponds exactly to s_5 in the consistent reference plan $\mathcal{A}_1$ (cf. Figure 2), meaning the remainder of the plan is executable. Thus, $\mathcal{I}_{\mathrm{e}}(\mathcal{P}_1, \mathcal{A}_2) = 2$.*

While the basic error-propagation measure $\mathcal{I}_{\mathrm{e}}$ captures whether or not an action violates its preconditions, it treats all such violations equally, regardless of their severity. However, in practice, some steps may violate just one precondition, while others violate several, potentially indicating a more severe inconsistency. To account for this, we introduce a precondition-based refinement of the measure. Instead of counting only the number of invalid actions, this version, denoted by $\mathcal{I}_{\mathrm{e}}^{\mathrm{p}}$, quantifies the total number of unsatisfied preconditions throughout the plan execution. This allows for a finer-grained assessment of how severely a plan deviates from consistency.

Definition 6. *The* precondition-based *error-propagation inconsistency measure $\mathcal{I}_{\mathrm{e}}^{\mathrm{p}}$ is defined as*

$$\mathcal{I}_{\mathrm{e}}^{\mathrm{p}}(\mathcal{P}, \mathcal{A}) = \sum_{i=0}^{|\mathcal{A}|-1} |\mathrm{Pre}(a_i) \setminus s_i|$$

w.r.t. a given planning problem $\mathcal{P} = (\Sigma, s_0, g)$, with $\Sigma = (S, A, \tilde{\gamma}, L, O)$ being a planning domain, a given plan $\mathcal{A} = \langle a_0, \ldots, a_{n-1} \rangle$, and $s_{i+1} = \tilde{\gamma}(s_i, a_i)$.

Example 6. *From Example 5 we know that w.r.t. $\mathcal{A}_2$ and $\mathcal{P}_1$ from Example 1, there is one precondition violated w.r.t. a_1 and two w.r.t. a_3. Thus, $\mathcal{I}_{\mathrm{e}}^{\mathrm{p}}(\mathcal{P}_1, \mathcal{A}_2) = 1 + 2 = 3$.*

Repair-Based Inconsistency Measures In many applications, it may be desirable to not only assess how inconsistent a plan is, but also how easily repairable it would be. To this end, we define a repair-based inconsistency measure, which reflects the minimum number of modifications needed to transform a given plan into an executable one. We restrict our notion of repair to the basic structural operations of *action deletion* and *action insertion*. This reflects the practical scenario where a robot or a planning system may discard incorrect steps or add necessary ones without re-generating a plan from scratch.

Let $\mathcal{A} = \langle a_0, \ldots, a_{n-1} \rangle$ be a plan. We define a repair function $r(\mathrm{op}, a, i, \mathcal{A})$, with $\mathrm{op} \in \{\mathrm{del}, \mathrm{ins}\}$ and a, i with $a_i \in \mathcal{A}$, such that $r(\mathrm{op}, a, i, \mathcal{A}) = \langle a_0, \ldots, a_{i-1}, a_{i+1}, \ldots, a_{n-1} \rangle$ if $\mathrm{op} = \mathrm{del}$ and $r(\mathrm{op}, a, i, \mathcal{A}) = \langle a_0, \ldots, a_{i-1}, a, a_{i+1}, \ldots, a_{n-1} \rangle$ if $\mathrm{op} = \mathrm{ins}$. Further, we define $\mathcal{R}$ as a sequence of repairs $r_1, \ldots, r_m$, where each r_j is a repair operation of the form $r(\mathrm{op}_j, a_j, i_j, \mathcal{A})$, with $j \in \{1, \ldots, m\}$. The result of applying $\mathcal{R}$ to $\mathcal{A}$ is inductively defined as $\mathcal{A}_0 := \mathcal{A}$, and $\mathcal{A}_j := r(\mathrm{op}_j, a_j, i_j, \mathcal{A}_{j-1})$, such that $r(\mathcal{A}, \mathcal{R}) = \mathcal{A}_m$.

Definition 7. *Let $\mathcal{R}$ be a sequence of repairs. The* repair-based inconsistency measure *$\mathcal{I}_{\mathrm{r}}$ is defined as*

$$\mathcal{I}_{\mathrm{r}}(\mathcal{P}, \mathcal{A}) = \min\{|\mathcal{R}| \mid r(\mathcal{A}, \mathcal{R}) \textit{ is consistent w.r.t. } \mathcal{P}\}$$

w.r.t. a given planning problem $\mathcal{P} = (\Sigma, s_0, g)$, with $\Sigma = (S, A, \gamma, L, O)$ being a planning domain, a given plan $\mathcal{A} = \langle a_0, \ldots, a_{n-1} \rangle$, and $s_{i+1} = \gamma(s_i, a_i)$.

Example 7. *Consider again $\mathcal{A}_2$ from Example 1. We already know that inserting the action $\mathtt{stackT}(A)$ after the first action ($a_0 = \mathtt{unstack}(A, B)$), i. e., applying $\mathcal{R} = \langle r_1 \rangle$ with $r_1 = r(\mathrm{ins}, \mathtt{stackT}(A), 1, \mathcal{A}_2)$, results in a consistent plan. Hence, $\mathcal{I}_{\mathrm{r}}(\mathcal{P}_1, \mathcal{A}_2) = 1$.*

Note that several variations of the repair-based measure are possible. For instance, once could restrict the repair operation to deletions and therefore measure the minimum number of *deletions* necessary in order to make a given plan $\mathcal{A}$ consistent.

4 Detecting and Explaining Inconsistencies

The inconsistency measures introduced in the previous section serve the purpose of allowing for the quantification of inconsistency in plans, which can be useful, e. g., when comparing different learning-based/approximate planning approaches, or when evaluating the output quality of different LLMs within the same framework. However, these numerical values can also serve as a starting point for a more qualitative analysis. More specifically, in order to compute the measures introduced in the previous section, it is typically necessary to identify where inconsistencies occur within a given plan. This makes it possible to extract the sources of inconsistency, which can then be reused for generating human-understandable explanations of what went wrong.

Such explanations are particularly valuable in cognitive robotics, where interpretability is crucial. This kind of inconsistency analysis, however, requires exact algorithmic approaches—approximate methods may be faster, but they generally do not provide insight into the specific causes of inconsistency. Recent algorithmic work on inconsistency measurement in other areas, such as propositional logic (Niskanen et al. 2023; Kuhlmann et al. 2025) and linear temporal logic (Corea et al. 2024), has demonstrated approaches based on Boolean and maximum satisfiability solving, as well as answer set programming. Such approaches would in fact allow for the extraction of inconsistencies.

To illustrate the idea of extracting and explaining inconsistencies in more detail, we now turn to a concrete example based on the first-conflict measure.

Example 8. *Consider again Example 3. To compute the value of $\mathcal{I}_1(\mathcal{P}_1, \mathcal{A}_2)$, we need to find the index i of the first non-executable action. In this example, we can extract that the index we are aiming to find is 1, and we can also extract the corresponding action $a_1 = \mathtt{unstack}(B, C)$ and state s_1. Using this information, we can generate an explanation of the form "*The given plan $\mathcal{A}$ is inconsistent, as action a_i is not executable, because the corresponding state s_i does not satisfy the preconditions $\mathrm{Pre}(a_i)$ of a_i*". W.r.t. $\mathcal{A}_2$, we would replace $\mathcal{A}$ by $\mathcal{A}_2$, a_i by a_1, s_i by s_1, and $\mathrm{Pre}(a_i)$ by $\{\mathtt{on}(B, C), \mathtt{clear}(B), \mathtt{gripperEmpty}\}$. Likewise, based on the computation of $\mathcal{I}_1^{\mathrm{p}}$, we can extend the explanation by providing the concrete set of violated preconditions.*

Furthermore, we argue that applying multiple measures on the same plan can offer complementary perspectives on inconsistency. For instance, the first-conflict measure provides a clear diagnostic of where executability first fails, while the error-propagation measure offers a more global view by reflecting how far-reaching the consequences of early failures are.

5 Conclusions

Learning-based methods are increasingly used in robotics and cognitive systems for planning tasks, however, they do not guarantee correct outputs. To quantitatively assess potential failures, this paper introduces the notion of inconsistency measurement for planning, focusing on plan-level inconsistencies, and, in particular, non-executable plans. We proposed several inconsistency measures that capture the degree of inconsistency in a given (non-executable) plan and illustrated them through formal definitions and examples. In addition, we discussed how the detection of inconsistencies can serve as a basis for generating explanations.

This work opens several directions for future research, e. g., the development of algorithmic approaches and their empirical evaluation; the design of further inconsistency measures; the extension to other types of planning inconsistencies (e. g., on the domain- or task-level); and the application to more expressive planning formalisms beyond the classical setting used in this work.

References

Aineto, D.; Jiménez, S.; and Onaindia, E. 2018. Learning strips action models with classical planning. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 28, 399–407.

Ammoura, M.; Raddaoui, B.; Salhi, Y.; and Oukacha, B. 2015. On measuring inconsistency using maximal consistent sets. In *European Conference on Symbolic and Quantitative Approaches to Reasoning and Uncertainty*, 267–276. Springer.

Asai, M., and Fukunaga, A. 2018. Classical planning in deep latent space: Bridging the subsymbolic-symbolic boundary. In *Proceedings of the aaai conference on artificial intelligence*, volume 32.

Baum, J.; Nicholson, A. E.; and Dix, T. I. 2012. Proximity-based non-uniform abstractions for approximate planning. *J. Artif. Intell. Res.* 43:477–522.

Bertossi, L. 2018. Measuring and computing database inconsistency via repairs. In *Proceedings of the 12th International Conference on Scalable Uncertainty Management (SUM'18)*.

Besnard, P., and Grant, J. 2020. Relative inconsistency measures. *Artificial Intelligence* 280:103231.

Capitanelli, A., and Mastrogiovanni, F. 2024. A framework for neurosymbolic robot action planning using large language models. *Frontiers in Neurorobotics* 18:1342786.

Chu, K.; Zhao, X.; Weber, C.; and Wermter, S. 2025. LLM+MAP: Bimanual robot task planning using large language models and planning domain definition language. *arXiv preprint arXiv:2503.17309*.

Corea, C.; Kuhlmann, I.; Thimm, M.; and Grant, J. 2024. Paraconsistent reasoning for inconsistency measurement in declarative process specifications. *Information Systems* 122:102347.

Corea, C.; Grant, J.; and Thimm, M. 2022. Measuring inconsistency in declarative process specifications. In *20th International Conference on Business Process Management*, 289–306. Springer.

Corea, C.; Thimm, M.; and Delfmann, P. 2021. Measuring inconsistency over sequences of business rule cases. In Bienvenu, M.; Lakemeyer, G.; and Erdem, E., eds., *Proceedings of the 18th International Conference on Principles of Knowledge Representation and Reasoning (KR'21)*, 656–660. IJCAI Organization.

Das, D.; Banerjee, S.; and Chernova, S. 2021. Explainable AI for robot failures: Generating explanations that improve user assistance in fault recovery. In *Proceedings of the 2021 ACM/IEEE international conference on human-robot interaction*, 351–360.

Decker, H., and Misra, S. 2017. Database inconsistency measures and their applications. In *Proceedings of the 23rd International Conference on Information and Software Technologies (ICIST 2017)*.

Elnoor, M.; Weerakoon, K.; Seneviratne, G.; Xian, R.; Guan, T.; Jaffar, M. K. M.; Rajagopal, V.; and Manocha, D. 2024. Robot navigation using physically grounded vision-language models in outdoor environments. *arXiv preprint arXiv:2409.20445*.

Eriksson, S.; Röger, G.; and Helmert, M. 2018. A proof system for unsolvable planning tasks. In *Proceedings of the international conference on automated planning and scheduling*, volume 28, 65–73.

Fox, M.; Gerevini, A.; Long, D.; Serina, I.; et al. 2006. Plan stability: Replanning versus plan repair. In *ICAPS*, volume 6, 212–221.

Ghallab, M.; Nau, D.; and Traverso, P. 2004. *Automated Planning: theory and practice*. Elsevier.

Grant, J., and Hunter, A. 2011. Measuring consistency gain and information loss in stepwise inconsistency resolution. In *Proceedings of the 11th European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty (ECSQARU'11)*, 362–373. Springer.

Grant, J., and Hunter, A. 2013. Distance-based measures of inconsistency. In *European Conference on Symbolic and Quantitative Approaches to Reasoning and Uncertainty*, 230–241. Springer.

Grant, J., and Martinez, M. V., eds. 2018. *Measuring Inconsistency in Information*, volume 73 of *Studies in Logic*. College Publications.

Grant, J., and Parisi, F. 2024. On measuring inconsistency in graph databases with regular path constraints. *Artificial Intelligence* 335:104197.

Grant, J. 1978. Classifications for inconsistent theories. *Notre Dame Journal of Formal Logic* 19(3):435–444.

Heusner, M.; Wehrle, M.; Pommerening, F.; and Helmert, M. 2014. Under-approximation refinement for classical planning. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 24, 365–369.

Huang, W.; Abbeel, P.; Pathak, D.; and Mordatch, I. 2022. Language models as zero-shot planners: Extracting actionable knowledge for embodied agents. In *International conference on machine learning*, 9118–9147. PMLR.

Hunter, A., and Konieczny, S. 2008. Measuring inconsistency through minimal inconsistent sets. In *Proceedings of the 11th International Conference on Principles of Knowledge Representation and Reasoning (KR'08)*, 358–366. AAAI Press.

Hunter, A., and Summerton, R. 2006. A knowledge-based approach to merging information. *Knowledge-Based Systems* 19(8):647–674.

Hunter, A. 2006. How to act on inconsistent news: Ignore, resolve, or reject. *Data & Knowledge Engineering* 57(3):221–239.

Jabbour, S.; Ma, Y.; and Raddaoui, B. 2014. Inconsistency measurement thanks to MUS decomposition. In *Proceedings of the 2014 International Conference on Autonomous Agents and Multi-Agent Systems*, 877–884.

Khalastchi, E., and Kalech, M. 2018. On fault detection and diagnosis in robotic systems. *ACM Comput. Surv.* 51(1):9:1–9:24.

Knight, K. 2002. Measuring inconsistency. *Journal of Philosophical Logic* 31(1):77–98.

Kuhlmann, I.; Gessler, A.; Laszlo, V.; and Thimm, M. 2025. Comparison of sat-based and asp-based algorithms for inconsistency measurement. *Journal of Artificial Intelligence Research* 82:563–685.

Ma, Y.; Qi, G.; Xiao, G.; Hitzler, P.; and Lin, Z. 2009. An Anytime Algorithm for Computing Inconsistency Measurement. In *3rd International Conference on Knowledge Science, Engineering and management*, 29–40.

Martinez, A. B. B.; Arias, J. J. P.; and Vilas, A. F. 2004. On measuring levels of inconsistency in multi-perspective requirements specifications. In *Proceedings of the 1st Conference on the Principles of Software Engineering (PRISE'04)*.

Niskanen, A.; Kuhlmann, I.; Thimm, M.; and Järvisalo, M. 2023. Maxsat-based inconsistency measurement. In *ECAI 2023*. IOS Press. 1779–1786.

Parisi, F., and Grant, J. 2020. On measuring inconsistency in relational databases with denial constraints. In *Proceedings of the 24th European Conference on Artificial Intelligence (ECAI'20)*. IOS Press. 857–864.

Parisi, F., and Grant, J. 2021. On database inconsistency measures. In *29th Italian Symposium on Advanced Database Systems, SEBD*, volume 2994 of *CEUR Workshop Proceedings*, 200–208.

Parisi, F., and Grant, J. 2023. Relative inconsistency measures for indefinite databases with denial constraints. In *Proceedings of the 32nd International Joint Conference on Artificial Intelligence, IJCAI 2023, 19th-25th August 2023, Macao, SAR, China*, 3321–3329. ijcai.org.

Potyka, N., and Thimm, M. 2017. Inconsistency-tolerant reasoning over linear probabilistic knowledge bases. *International Journal of Approximate Reasoning* 88:209–236.

Rajvanshi, A.; Sikka, K.; Lin, X.; Lee, B.; Chiu, H.-P.; and Velasquez, A. 2024. Saynav: Grounding large language models for dynamic planning to navigation in new environments. In *International Conference on Automated Planning and Scheduling (ICAPS)*.

Rana, K.; Haviland, J.; Garg, S.; Abou-Chakra, J.; Reid, I.; and Suenderhauf, N. 2023. Sayplan: Grounding large language models using 3d scene graphs for scalable robot task planning. *arXiv preprint arXiv:2307.06135*.

Sarwar, M. M. S.; Ray, R.; and Banerjee, A. 2023. Explaining unsolvability of planning problems in hybrid systems with model reconciliation. In *Proceedings of the 21st ACM-IEEE International Conference on Formal Methods and Models for System Design*, 47–58.

Sathyamoorthy, A. J.; Weerakoon, K.; Elnoor, M.; Zore, A.; Ichter, B.; Xia, F.; Tan, J.; Yu, W.; and Manocha, D. 2024. Convoi: Context-aware navigation using vision language models in outdoor and indoor environments. In *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 13837–13844. IEEE.

Silver, T.; Chitnis, R.; Tenenbaum, J.; Kaelbling, L. P.; and Lozano-Pérez, T. 2021. Learning symbolic operators for task and motion planning. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 3182–3189. IEEE.

Silver, T.; Hariprasad, V.; Shuttleworth, R. S.; Kumar, N.; Lozano-Pérez, T.; and Kaelbling, L. P. 2022. Pddl planning with pretrained large language models. In *NeurIPS 2022 foundation models for decision making workshop*.

Slaney, J., and Thiébaux, S. 2001. Blocks world revisited. *Artificial Intelligence* 125(1-2):119–153.

Song, C. H.; Wu, J.; Washington, C.; Sadler, B. M.; Chao, W.-L.; and Su, Y. 2023. Llm-planner: Few-shot grounded planning for embodied agents with large language models. In *Proceedings of the IEEE/CVF international conference on computer vision*, 2998–3009.

Sreedharan, S.; Srivastava, S.; Smith, D. E.; and Kambhampati, S. 2019. Why can't you do that HAL? explaining unsolvability of planning tasks. In Kraus, S., ed., *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019*, 1422–1430. ijcai.org.

Thimm, M. 2016. Stream-based inconsistency measurement. *International Journal of Approximate Reasoning* 68:68–87.

Thimm, M. 2018. On the evaluation of inconsistency measures. *Measuring Inconsistency in Information* 73:19–60.

Thimm, M. 2019. Inconsistency measurement. In Amor, N. B.; Quost, B.; and Theobald, M., eds., *Proceedings of the 13th International Conference on Scalable Uncertainty Management (SUM'19)*, volume 11940 of *Lecture Notes in Artificial Intelligence*, 9–23. Springer International Publishing.

Ulbricht, M.; Thimm, M.; and Brewka, G. 2016. Measuring inconsistency in answer set programs. In *Logics in Artificial Intelligence: 15th European Conference, JELIA 2016, Larnaca, Cyprus, November 9-11, 2016, Proceedings 15*, 577–583. Springer.

Van Der Krogt, R., and De Weerdt, M. 2005. Plan repair as an extension of planning. In *ICAPS*, volume 5, 161–170.

Wang, R.; Yang, Z.; Zhao, Z.; Tong, X.; Hong, Z.; and Qian, K. 2024. LLM-based robot task planning with exceptional handling for general purpose service robots. In *2024 43rd Chinese Control Conference (CCC)*, 4439–4444. IEEE.

Xiao, G., and Ma, Y. 2012. Inconsistency measurement based on variables in minimal unsatisfiable subsets. In *Proceedings of the 20th European Conference on Artificial Intelligence (ECAI'12)*. IOS Press. 864–869.

Zhang, X.; Wang, K.; Wang, Z.; Ma, Y.; Qi, G.; and Feng, Z. 2017. A distance-based framework for inconsistency-tolerant reasoning and inconsistency measurement in dl-lite. *International Journal of Approximate Reasoning* 89:58–79.

Zhou, L.; Huang, H.; Qi, G.; Ma, Y.; Huang, Z.; and Qu, Y. 2009. Measuring inconsistency in DL-Lite ontologies. In *2009 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology*, volume 1, 349–356. IEEE.

Zhu, X., and Jin, Z. 2005. Inconsistency measurement of software requirements specifications: An ontology-based approach. In *10th IEEE International Conference on Engineering of Complex Computer Systems (ICECCS'05)*, 402–410. IEEE.