On the Use of Large Language Models as Domain Model Configurators

Ilche Georgievski[1], **Daniel Elis**[1], **Mauro Vallati**[2]

[1]Service Computing Department, IAAS, University of Stuttgart, Stuttgart, Germany
[2]School of Computing and Engineering, University of Huddersfield, Huddersfield, United Kingdom
ilche.georgievski@iaas.uni-stuttgart.de, danielelis@web.de, m.vallati@hud.ac.uk

Abstract

The quality of domain models is critical for efficient plan generation in domain-independent planning. Domain model configuration has emerged as a means to influence planner behaviour without altering the planner's internals. Prior configuration approaches are either manual or require extensive iterative planner evaluations tied to a specific planner. We investigate whether LLMs can serve as domain configurators via a single forward pass, without planner feedback during configuration. We present an automated, planner-agnostic approach that produces structural domain variants guided by domain-engineering principles, together with a validation pipeline that filters variants preserving element sets of the original domain. We adopt a planner-agnostic setting, not expecting a single configuration to suit all planners, but, as a baseline, isolating what LLMs contribute without feedback and serving as a generator for studying planner sensitivity at scale. We conduct a systematic evaluation across seven LLMs, five prompting strategies, five planners, and five domains. We find that LLM choice is the primary determinant of configuration quality, with minimal zero-shot prompting outperforming richer strategies, and that the resulting variants elicit systematic but heterogeneous planner sensitivity. We discuss the economics and biases of LLM-based configuration as well as implications for planner evaluation and domain engineering.

Introduction

Domain-independent planning relies on the modular separation between a planning engine and a domain model that defines its elements, such as actions and predicates. This modularity enables planning engines to accept different domain models, depending on the application, enabling flexible use of planning technology across diverse real-world settings. It also creates an opportunity: if a domain model can be automatically transformed into a representation more efficient for planning engines to process, planning performance may improve without altering the engines. The relevance of this opportunity extends well beyond performance optimisation. Domain models are not necessarily static artefacts: they may evolve through refactorings during development, they may be restructured by different domain modellers, and they can be increasingly produced by automated pipelines that emit varied outputs for the same underlying specification. Understanding which domain model variations planners tolerate and which systematically alter their behaviour therefore directly impacts reliability, reproducibility, and user trust when planning is deployed in real systems.

This idea underlies a broad family of reformulation and configuration techniques that generate alternative representations of a domain or instance to increase planning efficiency. Reformulation methods, including macro-operators (Vallati, Chrpa, and Serina 2020), action-schema splitting (Areces et al. 2014), and entanglement reduction (Chrpa and Barták 2009; Chrpa, Vallati, and McCluskey 2019), transform a domain model into a representation that can accelerate planning. Domain model configuration focuses on modifying the structural arrangement of domain models, such as the ordering of predicates and actions. Early work has observed that even minor structural changes to domains specified in the Planning Domain Definition Language (PDDL) (McDermott et al. 1998) can substantially influence planner performance and even change IPC rankings (Howe and Dahlman 2002; Riddle, Holte, and Barley 2011). Other works operationalise this effect more directly, using learning to select promising configurations (Vallati et al. 2015) or heuristics to order actions (Vallati et al. 2021). They depend on planner evaluations, requiring numerous runs, and operate within a relatively narrow configuration space. In this body of work, sensitivity to structural variation has been documented across multiple planner generations, indicating a phenomenon rather than a transient implementation concern of a single planner.

Advances in Large Language Models (LLMs) suggest a new configuration paradigm. While LLMs can generate and repair PDDL models from text and validator feedback (Guan et al. 2023; Oswald et al. 2024; Huang and Zhang 2025; Casciani et al. 2025), they have been shown to perform non-trivial edits and refactoring of code-like artefacts based on natural-language instructions (Cassano et al. 2023). This suggests that they can be used to restructure domain model specifications, potentially without requiring explicit training or planner evaluations. The advantages are twofold. On the efficiency side, LLM-based configuration offers reduced computational overhead, planner-agnostic applicability, and zero-shot deployment without task-specific training. On the characterisation side, it makes it practical to generate domain model variants at scale, which is what studying planner sensitivity requires. These advantages come with caveats, such as LLMs may hallucinate or violate stated constraints.

We explore whether LLMs can serve as domain model configurators along two complementary axes: (i) *generation*, i.e., whether LLMs can produce valid, semantically equivalent structural variants of PDDL domains without planner feedback and which factors determine success (validity and regression analysis), and (ii) *characterisation*, i.e., what patterns these variants exhibit (change-frequency analysis) and how planners respond to them (performance gains and their distributions). Concretely, we prompt LLMs with domain-engineering principles to generate candidate configurations, filter them through a validation pipeline that checks syntactic validity and element-set preservation, and evaluate those that pass across seven LLMs, five prompting strategies, five planners, and five domains.

Beyond demonstrating feasibility, this work (i) introduces the first planner-agnostic, single-forward-pass framework for LLM-based domain model configuration; (ii) provides a systematic investigation of LLMs for restructuring PDDL domain models, extending beyond existing work on PDDL generation and repair; (iii) empirically characterise both the degree of semantic equivalence preserved by LLM-generated domain variants and the structural modification patterns they introduce; (iv) demonstrates that configuration effectiveness is highly planner-dependent, with heterogeneous sensitivity observable across planners and domains; (v) shows that LLM choice dominates prompting strategy and decoding randomness as the primary factor influencing configuration quality; and (vi) derives practical implications for domain engineering, identifying where LLM-driven configuration is effective and where it introduces risks.

Background

Planning While we rely on PDDL to represent domain models, we use the following formal definitions to facilitate the discussion of the underlying formalism. Let F be a finite set of predicates that describe the world and F_g the corresponding set of ground predicates. A state $s \subseteq F_g$ is a set of ground predicates representing all facts that hold in the world. An action schema a is a tuple $\langle par(a), pre(a), del(a), add(a)\rangle$, where $par(a)$ is a set of variables occurring in a, $pre(a) \subseteq F$ is a set of predicates that must hold to apply the action, $del(a) \subseteq F$ is a set of predicates that become false, and $add(a) \subseteq F$ is a set of predicates that become true after the action is applied. Actions are ground instances of action schemas. An action $a_g = \langle pre_g(a_g), del_g(a_g), add_g(a_g)\rangle$ is applicable in state s if and only if $pre_g \subseteq s$. Application of a_g in s results in a state $s' = (s \setminus del_g(a_g)) \cup add_g(a_g)$. A domain model D is a pair $\langle F, A\rangle$, where A is a set of action schemas. A planning task is a pair $T = \langle D_T, P_T\rangle$, where D_T is a domain model and P_T is a triple consisting of F_g, an initial state s_I, and a set of goal ground predicates s_G. A solution plan of T is a sequence of actions such that a consecutive application of plan actions starting in s_I results in a state satisfying s_G. We denote $Plans(D_T, P_T)$ as the set of all solution plans for the problem instance P_T when evaluated under domain D_T.

Domain Model Configuration A domain model is a central artefact of knowledge engineering for automated planning. In practice, its construction is often ad hoc, and the modelling choices made by engineers can substantially influence planner performance (McCluskey, Vaquero, and Vallati 2017; Georgievski 2023). Although the formal definition of domain models treats predicates, action schemas, and their internal components as unordered sets, their ordering in concrete PDDL descriptions can significantly affect the behaviour of planning engines (Howe and Dahlman 2002). To make this explicit, we use the following formalism, in line with Vallati et al. (2021). Let $D = \langle F, A\rangle$ be a domain model. We equip F and A with total order relations $\preceq_F$ and $\preceq_A$, respectively. For each action schema $a \in A$, we also define total order relations $\preceq_{par(a)}$ over $par(a)$, $\preceq_{pre(a)}$ over $pre(a)$, and $\preceq_{\mathit{eff}(a)}$ over $\mathit{eff}(a) = \{del(a) \cup add(a)\}$. For simplicity, let $\preceq$ denote the set of all these relations. Then, we say that $D_{\preceq}$ is a domain model configuration of D obtained by writing the domain using the orders in $\preceq$. The domain model configuration space $C(D)$ consists of all possible domain model configurations, i.e., all possible choices of $\preceq$. Its cardinality is determined by the number of permutations of each ordered set: $|C(D)| = |F|!\,|A|!\prod_{a\in A}(|par(a)|!\,|pre(a)|!\,|\mathit{eff}(a)|!)$, which grows factorially in the size of the domain. Even for relatively small models, $|C(D)|$ becomes very large, making exhaustive exploration infeasible and motivating the need for techniques that can identify promising domain variants.

LLMs LLMs are probabilistic next-token predictors trained on large text corpora. Given a token sequence, an LLM outputs a probability distribution over possible continuations, and a decoding strategy selects concrete tokens from this distribution. Modern LLMs rely almost exclusively on the Transformer architecture (Vaswani et al. 2017) and operate within a fixed context window, limiting the number of tokens they can condition on during generation. LLM behaviour is primarily controlled through prompts, which specify the task or transformation to be performed, and temperature, a decoding parameter that regulates sampling stochasticity: low temperatures yield more deterministic outputs, whereas higher temperatures introduce variability and diversity. Different LLMs are trained on various data mixtures and can be tuned for specialised tasks (e.g., code generation). Surveys of model families and capabilities can be found in Zhao et al. (2023); Minaee et al. (2024).

LLM-Based Domain Model Configuration

We now turn to characterising the task we are trying to solve and the approach we use to solve it.

Definition 1 (LLM-Based Domain Model Configuration)**.** *Let $D = \langle F, A\rangle$ be a domain model and let $C(D)$ denote the configuration space of D, consisting of all domain variants obtained by imposing total orderings over predicates, actions, parameters, preconditions, and effects. An* LLM-based domain configuration function *is a mapping $f_{LLM} : D \to D'$, which, given an original domain D, produces a candidate domain $D' = f_{LLM}(D)$. The candidate D' is a valid configuration of D, denoted $D_{\preceq}$, if and only if it satisfies the following three conditions:*

1. *Structural transformation: The ordering relations differ from the original:* $\exists X \in \{F, A, pre(a), \mathit{eff}(a)\} : \preceq_{X,D_\preceq} \neq \preceq_{X,D}$.
2. *Syntactic validity:* $D_\preceq$ *is a well-formed PDDL domain.*
3. *Semantic equivalence:* $D_\preceq \in C(D)$. *This implies* $Plans(D, P) = Plans(D_\preceq, P)$ *for every problem instance* P *over* D.

Prompt Design

We aim to explore LLMs' capabilities to predict beneficial domain model configurations without direct feedback from planners. Since LLMs may not have inherent knowledge of PDDL semantics or planning conventions, we need to enforce an LLM model to internalise domain-independent configuration principles, such as predicate placement conventions, action ordering heuristics, and precondition structuring strategies, and apply them appropriately based solely on syntactic analysis. To achieve this, we operationalise the configuration function f_{LLM} by a prompt $\mathcal{T}$. The prompt design thus directly determines the model's ability to generate valid, efficiency-oriented domain model configurations.

Effective prompt design for domain model configuration must address three fundamental challenges. First, an LLM must understand its role and expertise level to interpret technical context, such as PDDL syntax and planning terminology, appropriately. Second, strict constraints must prevent the LLM from making semantic modifications, such as additions, deletions, or logical alterations. Third, the LLM requires explicit or implicit guidance on what constitutes beneficial reordering, since optimal domain model configurations can depend on planner-specific heuristics and search strategies that are not observable from domain syntax. These requirements motivate a structured prompt design where role assignment establishes appropriate context, constraints enforce a structural focus or semantic preservation, and variable guidance mechanisms provide configuration knowledge at different levels of specificity.

To isolate the effect of different guidance mechanisms and evaluate whether declarative rules, demonstrations, or reasoning protocols improve domain model configuration quality, it is essential that the employed prompting strategies share a common foundation. To that end, we define a common prompt template $\mathcal{T}$ that consists of four components: $\mathcal{T} = \langle r, C, D, G \rangle$, where r is the role assignment establishing the model as a PDDL domain engineering expert, C is the constraint set explicitly prohibiting semantic modifications, renaming, or formatting changes, D is the input PDDL domain to be configured, and G is the guidance component encoding configuration knowledge.

To define the guidance mechanisms, we resort to the principal prompting strategies employed in planning research (Kojima et al. 2023; Valmeekam et al. 2022; Wang et al. 2022): zero-shot prompting, few-shot prompting, and chain-of-thought prompting. Based on these principal prompting strategies, we define five guidance mechanisms formalised through the prompt template by progressively enriching the guidance component G.

To test a model's inherent understanding of planning conventions, we define a zero-shot-short prompt $\mathcal{T}_{\mathrm{ZS-S}}$ as a tuple $\langle r, C, D, G_{min} \rangle$, where G_{min} contains only a task instruction to reorder for efficiency while preserving semantics, without explicit guidance principles.

To test whether domain-engineering principles can guide valid domain model transformations, we define a zero-shot-long prompt $\mathcal{T}_{\mathrm{ZS-L}}$ as a tuple $\langle r, C, D, G_\preceq \rangle$, where $G_\preceq = G_{KE} \cup G_{PDDL}$ combines guidance principles drawn from both domain-model configuration literature and PDDL modelling practice. In particular, G_{KE} consists of the following domain-engineering configuration principles:

- g_1: Action ordering — prioritise actions with fewer preconditions or enabling subsequent actions (Franco et al. 2019; Vallati et al. 2021).
- g_2: Predicate grouping — position frequently-used or goal-relevant predicates early (Franco et al. 2019; Helmert 2009).
- g_3: Precondition ordering — order by restrictiveness for early failure detection (Vallati et al. 2021).
- g_4: Static predicate grouping — cluster predicates never modified by actions (Fox and Long 2003).

G_{PDDL} consists of two PDDL modelling standards:

- g_5: Effect ordering — place add-effects before delete-effects (Fox and Long 2003; Helmert 2009).
- g_6: Parameter consistency — maintain uniform parameter ordering across actions (McCluskey, Vaquero, and Vallati 2017).

As these principles may introduce ambiguity or overfitting, they alone may be insufficient to fully constrain the model's interpretation, particularly across varying domain structures. Adding illustrative examples can appear helpful. To test whether concrete demonstration can improve principle application, we define a few-shot-short prompt $\mathcal{T}_{\mathrm{FS-S}}$ as a tuple $\langle r, C, D, G_\preceq \cup E_1 \rangle$, where $E_1 = \langle D_{sub}, D_{opt}, A_1 \rangle$ is a single annotated example consisting of a suboptimal domain variant D_{sub}, its optimised counterpart D_{opt}, and annotations A_1 linking transformations to principles.

To test whether multiple examples increase coverage of domain model configuration patterns and potentially improving the generalisation across diverse domains, we define a few-shot-long prompt $\mathcal{T}_{\mathrm{FS-L}}$ as a tuple $\langle r, C, D, G_\preceq \cup \{E_1, E_2\} \rangle$, where $E_2 = \langle D'_{sub}, D'_{opt}, A_2 \rangle$ provides a second annotated example from a different domain.

Finally, we assess whether intermediate verbalisation can improve transformation quality. Chain-of-thought prompting has been shown to improve LLM performance on tasks requiring multi-step structured reasoning (Kojima et al. 2023; Wang et al. 2022). In our case, the rationale is that requiring the model to first articulate which principles apply to which domain elements may reduce unintended edits during the subsequent rewrite. We define a chain-of-thought prompt $\mathcal{T}_{\mathrm{CoT}}$ as a tuple $\langle r, C, D, G_\preceq \cup M \rangle$, where M implements a multi-turn interaction protocol consisting of analysis and execution phases. In the analysis phase, the model receives $\langle r, C, D, G_\preceq \rangle$ and produces an explicit reasoning strategy R that articulates planned transformations by referencing $G_\preceq$. In the execution phase, the model receives $\langle R, D \rangle$ and produces $D_\preceq$.

Algorithm 1: Configuration Validation Pipeline

Require: Original domain D, configured domain $D_{\preceq}$
Ensure: Validation result: VALID, INVALID, or REJECTED

1: **if** $\neg$ PARSE($D_{\preceq}$) **then**
2: **return** REJECTED {syntactic error}
3: **end if**
4: $D' \leftarrow$ NORMALISE(D)
5: $D'_{\preceq} \leftarrow$ NORMALISE($D_{\preceq}$)
6: **if** $D' = D'_{\preceq}$ **then**
7: **return** REJECTED {no effective transformation}
8: **end if**
9: $E \leftarrow$ {requirements, types, predicates, actions,
10: parameters, preconditions, add-effects, del-effects}
11: **for** each element type $e \in E$ **do**
12: $S_{D'} \leftarrow$ EXTRACTELEMENTS(D', e)
13: $S_{D'_{\preceq}} \leftarrow$ EXTRACTELEMENTS($D'_{\preceq}, e$)
14: **if** $S_{D'} \neq S_{D'_{\preceq}}$ as sets **then**
15: **return** INVALID {semantic changes detected}
16: **end if**
17: **end for**
18: **return** VALID {only syntactic reorderings}

Validation

Ensuring that LLM-configured domains are valid is critical for meaningful efficiency evaluation. We design a three-stage automated validation pipeline that filters valid domain model configurations, as shown in Algorithm 1. When the algorithm returns a valid domain model configuration, it means the configuration satisfies the three properties in Definition 1. Given that transformations are constrained to element reordering, we can validate semantic equivalence through structural comparison: verifying that all element sets remain unchanged. This check is both necessary and sufficient for plan equivalence, since identical element sets with potentially different orderings define the same state space and action applicability conditions.

A generated $D_{\preceq}$ undergoes sequential validation in three stages. In Stage 1 (Lines 1-3), $D_{\preceq}$ is parsed and rejected if parsing fails or $D_{\preceq}$ contains syntactic errors. In Stage 2 (Lines 4-8), D and $D_{\preceq}$ are normalised by removing comments and standardising S-expression formatting. We verify whether $D_{\preceq}$ differs from D; configurations identical to the original domain are rejected as they provide no information about the model's configuration capabilities. In Stage 3 (Lines 9-18), we perform an element-level check to ensure that no additions, deletions, or modifications occur at any structural level. We extract all top-level PDDL elements and an action's internal structure. For an element type, we build element sets and compare them. If the sets differ, a semantic change has occurred, making the configuration invalid.

Realisation and Experimental Setup

System Architecture Our proposed approach is realised by a modular, component-based system that requires a set of original PDDL domain and problem files, which serve as the input for both automated domain model configuration and baseline evaluation. All combinations of LLMs, prompt types, and temperatures are systematically instantiated. For each setting, a corresponding prompt is generated and passed to the LLM Interface, which provides a unified access for interaction with all selected models. The LLM Interface also encapsulates model-specific parameters, such as temperature and maximum token length. The LLM Interface interacts with an LLM to produce a domain model configuration. Each configured domain is forwarded to the Validator module for execution of the validation pipeline. The Planner Executor module runs selected planners on the original domain model and validated configurations. The system logs performance for each run. Finally, the Evaluation module calculates key performance metrics for each planner-domain combination and performs comparative analysis.

Prompts For the few-shot prompts $\mathcal{T}_{\mathrm{FS-S}}$ and $\mathcal{T}_{\mathrm{FS-L}}$, we use examples from two benchmark domains. In particular, the example for $\mathcal{T}_{\mathrm{FS-S}}$ is drawn from the IPC 2008 Elevator domain, which is selected as a representative sample that provides a coherent context in which all guidance principles can be applied meaningfully. The example for $\mathcal{T}_{\mathrm{FS-L}}$ is drawn from the Tetris domain, which is chosen due to its compact representation and documented sensitivity to domain model configuration (Vallati et al. 2021). The examples E_j, $j = \{1, 2\}$ are generated using GPT-4o: given $G_{\preceq}$ and an original domain model, GPT-4o produced paired configurations $\langle D_{sub}, D_{opt} \rangle$ with annotations A_i explaining each transformation. Examples of prompts, as well as D_{sub} and D_{opt} encodings, are provided in the supplementary material.

Domains We use five benchmarks: Barman, Genome Edit Distances, Thoughtful, Transport, and Visit All. These benchmarks are corrected domains and verified solvable instances from IPC 2014, obtained from (University of Potsdam 2017). We use 20 problem instances per domain.

Planners Our planner selection comprises five systems as detailed in Table 1. We select planners spanning different translation and search components. Notably, the FD translator appears in three planners, but at versions spanning over a decade, and SIW and Madagascar have different translators. This heterogeneity lets us observe whether sensitivity to structural variation persists across planning frameworks.

LLMs We use seven LLMs accessible via API, balancing proprietary and open-source models: GPT-4o (OpenAI 2024a), o4-mini (OpenAI 2024b), Gemini 1.5 Pro (Reid et al. 2024), Claude 3.5 Sonnet (Anthropic 2024), DeepSeek-Chat (Liu et al. 2024), Llama-3.1-8B-Instruct (Dubey et al. 2024), and Mixtral-8x7B-Instruct (Jiang et al. 2024). We test four temperature settings: 0.0 (deterministic, greedy sampling), 0.2 (stable outputs), 0.5 (balanced), and 0.7 (increased diversity) (Kelly et al. 2023; Mahdavi et al. 2024; Stein et al. 2025).

Implementation We implemented the system in Python, with planners and VAL containerised using Docker for reproducibility. Stage 1 validation uses VAL for syntactic checking. For each validated configuration, we generate structured difference reports documenting syntactic

Planner	Source	Version	Translator	Translator provenance
SIW (Lipovetzky et al. 2014)	LAPKT GitHub[a]	Commit `f58ee5c` (2025)	Tarski	Bundled with SIW (2019)
BFS_f (Lipovetzky et al. 2014)	LAPKT GitHub	Commit `f58ee5c` (2025)	FD translate	Standalone[b] Commit `d387b94` (2020)
FD (Röger, Pommerening, and Seipp 2014)	FD GitHub[c]	Commit `e02b0f8` (2025)	FD translate	Bundled with FD
Mercury (Katz and Hoffmann 2014)	Mercury GitHub[d]	Commit `6dd8755` (IPC 2014)	FD translate	Bundled with Mercury
Madagascar (Rintanen 2014)	Madagascar's page[e]	MpC executable (IPC 2014)	Native	Bundled with Madagascar

[a]https://github.com/LAPKT-dev/LAPKT-public/tree/Devel2.0/ [b]https://github.com/LAPKT-dev/fd_translate
[c]https://github.com/aibasel/downward [d]https://github.com/b-it-bots/mercury_planner [e]https://research.ics.aalto.fi/software/sat/madagascar/

Table 1: Planners used in our evaluation, with version and translator provenance.

changes, semantic changes, added/removed elements, and specific transformations applied. These reports support debugging and enable qualitative analysis of configuration patterns across prompting strategies. Results are exported as CSV files, tables, and visualisations. We conduct all experiments in a dedicated cloud-based environment: Google Cloud Platform, with a c2-standard-8 (8 vCPU, 32 GB RAM) instance running Ubuntu 20.04 LTS. For each container run, we enforce resource constraints of one CPU and eight GB of RAM. All artefacts are publicly available.[1]

Metrics and Analysis We assess configurations using (1) validity rates, (2) a logistic regression whose binary outcome is full validity and whose predictors are LLM, $\mathcal{T}$, and temperature, and (3) change frequency, i.e., how often a PDDL component is modified across candidates. Planner performance is evaluated via Coverage, PAR10, IPC Score, and Plan Cost, all expressed as gains relative to baseline, with positive values indicating improvement. Following Vallati et al. (2015), IPC Score Single compares each planner to its own best result across configurations, while IPC Score Multi compares to the best result across all planners. We use kernel density estimation to visualise the full distribution of gains per LLM and the Shapiro-Wilk test to assess normality. Given non-normal distributions, we apply the Friedman test (with mean ranks and Kendall's W) to compare LLMs, prompts, and temperatures, followed by post-hoc Nemenyi tests with Bonferroni correction when significant.

Results

Across the seven LLMs, four temperature settings, and five prompting strategies, we obtained 140 unique configurations per domain and 700 configurations across all five domains. Out of these, 514 (73%) passed syntactic validation, while 350 (50%) were semantically equivalent to their original domains. The intersection of these two sets comprises 343 domains that were evaluated with five planners on 20 problem instances each, yielding 34,300 planner execution results.

Configuration Validity Rates

Table 2 presents validity rates across prompting strategies, LLMs, and domains. $\mathcal{T}_{\mathrm{ZS-S}}$ achieves the highest rates, followed by $\mathcal{T}_{\mathrm{FS-L}}$. Other strategies range from 70–74% syntactic and 46–49% semantic validity. LLM performance varies substantially: GPT-4o and Gemini achieve 100% syntactic validity, but significantly differ in semantic equivalence (96% vs. 38%); Claude and DeepSeek show intermediate syntactic validity; other open-source models perform poorly. Domain variation is also evident: Visit All shows the highest semantic equivalence, Transport the highest syntactic validity, and Thoughtful the lowest rates; Barman and Genome Edit Distances fall between these extremes. Temperature has minimal variation: syntactic validity remains stable (72–74%), semantic equivalence varies slightly (47–53%), and full validity peaks 51% at temperature 0.7.

Category	SV (%)	SEV (%)	Both (%)
$\mathcal{T}_{\mathrm{ZS-S}}$	**79.29**	**55.00**	**55.00**
$\mathcal{T}_{\mathrm{ZS-L}}$	70.00	47.86	45.00
$\mathcal{T}_{\mathrm{FS-S}}$	72.14	46.43	46.43
$\mathcal{T}_{\mathrm{FS-L}}$	71.43	52.14	50.71
$\mathcal{T}_{\mathrm{CoT}}$	74.29	48.57	47.86
Claude	80.00	54.00	54.00
DeepSeek	76.00	56.00	56.00
Gemini	**100.00**	38.00	38.00
GPT-4o	**100.00**	**96.00**	**96.00**
Llama3	24.00	17.00	14.00
Mixtral	37.00	26.00	22.00
o4-mini	97.00	63.00	63.00
Barman	74.29	43.57	42.86
Genome Edit Distances	75.71	45.00	44.29
Thoughtful	50.71	20.71	20.71
Transport	**85.00**	60.71	57.86
Visit All	81.43	**80.00**	**79.29**

Table 2: Validity rates by prompts, LLMs, and domains. SV: Syntactic Validity; SEV: Semantic Equivalence Validity.

Logistic Regression

We conducted logistic regression on the binary outcome of producing a domain model that is both syntactically valid and semantically equivalent, using Claude and $\mathcal{T}_{\mathrm{ZS-S}}$ as references. Collinearity checks show no problematic dependencies, and all predictors have sufficient support. The logistic regression model converged successfully with a pseudo-R^2 of 0.22, indicating the predictors explain about 22% of the variance in the probability of producing a fully valid domain.

Table 3 presents coefficients ($\hat{\beta}$), standard errors, z-values, p-values, and 95% confidence intervals (CI). LLM

[1]https://github.com/PlanX-Universe/llms-domain-model-configurators/

Variable	$\hat{\beta}$	SE	z-value	p-value	95% CI
Intercept	0.43	0.29	1.47	0.142	[-0.14, 1.00]
$\mathcal{T}_{\mathrm{ZS-L}}$	**-0.54**	**0.28**	-1.94	**0.053**	**[-1.09, 0.01]**
$\mathcal{T}_{\mathrm{FS-S}}$	-0.46	**0.28**	-1.66	0.097	[-1.01, 0.08]
$\mathcal{T}_{\mathrm{FS-L}}$	-0.23	**0.28**	-0.83	0.406	[-0.78, 0.31]
$\mathcal{T}_{\mathrm{CoT}}$	-0.39	**0.28**	-1.39	0.166	[-0.93, 0.16]
DeepSeek	0.08	**0.29**	0.29	0.775	[-0.48, 0.64]
Gemini	-0.66	**0.29**	-2.27	0.023	**[-1.22, -0.09]**
GPT-4o	**3.03**	0.55	5.52	**<0.001**	[1.96, 4.11]
Llama3	-1.99	0.35	**-5.65**	**<0.001**	[-2.68, -1.30]
Mixtral	-1.44	0.32	-4.56	**<0.001**	[-2.06, -0.82]
o4-mini	0.38	**0.29**	1.30	0.195	[-0.19, 0.94]
LLM_Temp	0.17	0.33	0.51	0.611	[-0.48, 0.81]

Table 3: Logistic regression coefficients predicting configuration validity. The *Intercept* represents baseline log-odds for the reference categories (Claude and $\mathcal{T}_{\mathrm{ZS-S}}$).

choice substantially affects validity. GPT-4o shows strong positive effects, while Llama3 and Mixtral show strong negative effects. Gemini has moderate negative effects. DeepSeek and o4-mini do not differ significantly from Claude at $\alpha = 0.05$. The prompting strategy shows no significant effects (all $p > 0.05$), with all variants having negative but non-significant coefficients relative to baseline. Temperature also has no significant effect.

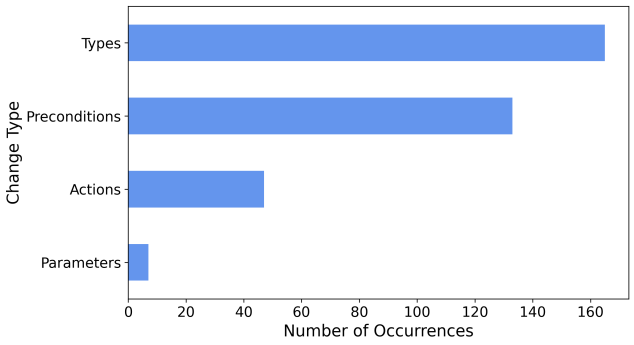
Frequency of Configuration Changes

Of 700 configurations, 106 (15%) were identical to originals, while 594 (85%) exhibited modifications: 350 (50%) contained semantic changes, 244 (35%) only syntactic changes. Figure 1 shows change distributions across components. Syntactic changes (Figure 1a) occur most in types (24%), followed by preconditions (19%), actions (7%), and parameters (1%). Semantic changes (Figure 1b) occur predominantly in actions (44%), types (22%), and requirements (21%). Internal components of actions show lower rates: preconditions with 2%, parameters with 0.7%, and predicates with 0.1%. The types component exhibits both high semantic and syntactic change rates, with 46% configurations altering types in some way.

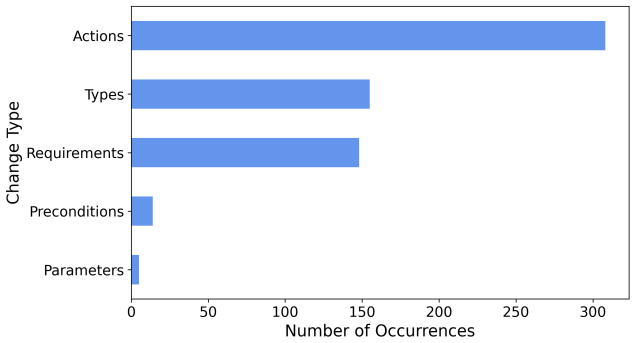
Planner Performance

We calculated the mean performance gain for each planner and evaluation metric relative to the original domains. Positive values represent improvements over originals. Table 4 presents mean gains per planner. SIW shows positive mean gains across all metrics, while FD shows modest positive gains. Madagascar, Mercury, and BFS_f show predominantly negative gains, with BFS_f showing the largest decreases. The last row shows that 14–26% of cases achieved positive gains, varying by metric.

Figure 2 shows IPC Score Single gains by planner-domain combination. SIW achieves positive gains across all domains. FD exhibits mixed behaviour, with a modest gain on Transport, a modest loss on Thoughtful, and no change on Barman, Genome Edit Distances, and Visit All. Madagascar shows gains on Visit All and Transport, with losses on Barman, Genome Edit Distances, and Thoughtful. Mercury and



(a) Frequency of syntactic changes by component type.



(b) Frequency of semantic changes by component type.

Figure 1: Distribution of semantic and syntactic changes.

BFS_f show predominantly negative gains, each with a single exception on Genome Edit Distances, and BFS_f shows the largest decreases overall.

Figure 3 shows kernel density estimates for IPC Score Single for each LLM. All distributions peak below zero (modes: -0.5 to -1.0), with most gains between -2 and +2. Llama3 and Mixtral show wider distributions, indicating greater variability. No LLM distribution peaks above zero.

Shapiro-Wilk tests indicate significant deviations from normality for all metrics, necessitating the use of non-parametric analysis. Table 5 presents Friedman test results. LLM choice shows significant effects for PAR10, IPC Score Multi (M), and Coverage, with o4-mini ranking best and Llama3 worst. Effect sizes are weak ($W \approx 0.2$). No significant effects occur for IPC Score Single (S) or Plan Cost. For prompts and temperatures, no statistically significant differences are observed for any metric ($p > 0.2$), and all effect sizes fell into the very weak range ($W < 0.07$). For each significant Friedman result, we performed pairwise Nemenyi post-hoc tests with Bonferroni correction. No pairwise comparisons reached statistical significance after correction, indicating that although global differences exist among LLMs for some metrics, no specific pairs differ significantly at the adjusted significance level. The supplementary material includes histograms, Q-Q plots, and normality test results.

Planner	Coverage Gain	PAR10 Gain	IPC Score Single Gain	IPC Score Multi Gain	Plan Cost Gain
SIW	0.068	205.59	0.588	0.313	51.97
FD	0.014	39.11	0.061	0.049	5.45
Madagascar	-0.023	-66.78	-0.154	-0.105	-9.50
Mercury	-0.121	-357.71	-1.064	-0.541	1.72
BFS_f	-0.162	-483.11	-3.897	-0.817	8.80
Positive Gain Proportion	0.14	0.25	0.26	0.26	0.15

Table 4: Mean performance gains per planner across all configurations. Last row: proportion of cases exceeding baseline.

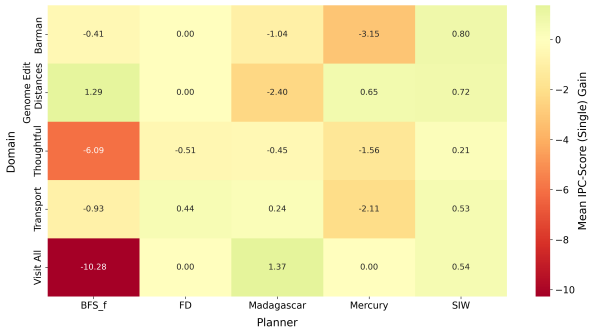


Figure 2: Mean IPC Score Single gain by planner and domain. Positive gain in green, negative gain in red.

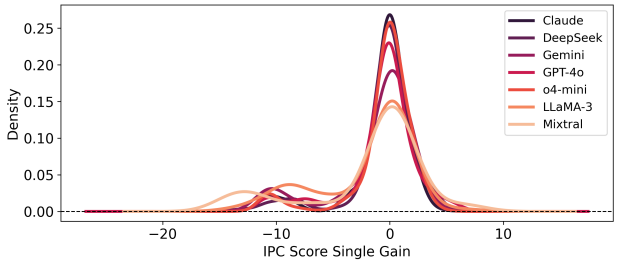


Figure 3: Kernel density estimates of IPC Score Single gains by LLM across all planner-domain combinations.

Discussion

Quality of LLM-Configured Domains The validity rates and logistic regression jointly indicate that LLM choice has a much larger influence on configuration quality than prompting strategy or temperature. Capable models (GPT-4o, o4-mini) produce near-complete valid configurations; intermediate models (Claude, DeepSeek, Gemini) produce partially valid configurations; smaller open-source models (Llama3, Mixtral) rarely do. The simplest prompting strategy, $\mathcal{T}_{\mathrm{ZS-S}}$, unexpectedly tends to produce the highest share of valid configurations, outperforming strategies with principles, examples, or chain-of-thought reasoning. Temperature has no detectable effect. Domain-level variation tracks complexity: Visit All and Transport yield higher validity rates than Thoughtful, suggesting that valid configuration depends both on LLM capacity and on how readily a domain exposes reordering opportunities that preserve element sets.

Var	Metric	p	SS	W	BMR	WMR
LLM	PAR10	0.009	Yes	0.22	o4-mini	Llama3
LLM	IPC Sc. (M)	0.015	Yes	0.21	o4-mini	Llama3
LLM	Coverage	0.018	Yes	0.20	o4-mini	Llama3
LLM	IPC Sc. (S)	0.078	No	0.15	o4-mini	Llama3
LLM	Plan Cost	0.482	No	0.12	Mixtral	Claude
Prom.	PAR10	0.229	No	0.05	$\mathcal{T}_{\mathrm{FS-S}}$	$\mathcal{T}_{\mathrm{ZS-S}}$
Prom.	IPC Sc. (M)	0.459	No	0.03	$\mathcal{T}_{\mathrm{FS-S}}$	$\mathcal{T}_{\mathrm{ZS-S}}$
Prom.	Coverage	0.200	No	0.05	$\mathcal{T}_{\mathrm{FS-S}}$	$\mathcal{T}_{\mathrm{ZS-S}}$
Prom.	IPC Sc. (S)	0.671	No	0.02	$\mathcal{T}_{\mathrm{FS-L}}$	$\mathcal{T}_{\mathrm{ZS-S}}$
Prom.	Plan Cost	0.255	No	0.07	$\mathcal{T}_{\mathrm{ZS-L}}$	$\mathcal{T}_{\mathrm{ZS-S}}$
Temp.	PAR10	0.754	No	0.01	0.7	0.2
Temp.	IPC Sc. (M)	0.727	No	0.01	0.7	0.2
Temp.	Coverage	0.782	No	0.01	0.0	0.2
Temp.	IPC Sc. (S)	0.727	No	0.01	0.0	0.2
Temp.	Plan Cost	0.907	No	0.01	0.7	0.0

Table 5: Friedman test results comparing LLM choice, prompts, and temperature across metrics (SS for statistically significant and BMR/WMR for best/worst mean ranks).

Cost and Feasibility A natural concern about LLM-based configuration is that a 50% valid-configuration rate may undermine its usefulness. The economics suggest the opposite. LLM generation takes seconds per domain, validation is purely structural and cheap, and frontier-model API costs are small per configuration. By contrast, SMAC-style configuration (Vallati et al. 2015) requires on the order of thousands of planner evaluations per promising configuration, tied to a specific planner. Even at a 50% validation rate, the cost of discarding invalid LLM outputs is small relative to a single planner evaluation. This makes it practical to generate and filter large pools of candidates per domain, enabling characterisation studies like ours and closed-loop extensions that iterate between generation and planner feedback. Valid configurations could also be sampled at random; the distinguishing value of LLMs is that their variants are non-uniform and principle-correlated, probing the kinds of structural variation human domain modellers introduce, and each variant is traceable to explicit guidance. A controlled comparison against random and heuristic reordering baselines remains important future work.

Configuration Patterns and LLM Biases Despite explicit instructions to preserve semantics, 50% of candidates are rejected because they modify element sets. The pattern reveals systematic biases in how LLMs interpret the restructuring task. Action-level modifications dominate (44% of candidates altered the action set, while only 7% reordered

actions without modification), indicating that LLMs find high-level structural changes easier than fine-grained reordering, which is the opposite of what the task requires. Particularly revealing are unintended modifications to types (22%) and requirements (21%), components the prompts never target, indicating that LLMs apply implicit restructuring patterns rather than adhering to explicit constraints. The inverse pattern holds for parameters, which are explicitly mentioned in prompts but reordered in only 1% of candidates. This disconnect pattern suggests that learned patterns dominate prompt instructions when the two conflict.

Planner Sensitivity to Configuration Planner responses are systematic but heterogeneous. SIW shows gains on all five domains. A plausible mechanism is SIW's width-based search: novelty computation depends on the order of predicate and action enumeration, so reconfiguring directly alters pruning. BFS_f and Mercury degrade on four of five domains each (both with a single exception on Genome Edit Distances), with BFS_f showing the largest magnitude losses; FD exhibits modest domain-dependent sensitivity with a gain on Transport and a loss on Thoughtful; Madagascar shows gains on two domains and losses on three. We treat each planner as a black box, characterising the integrated system rather than isolating component effects: a domain modeller interacts with a planner as distributed. FD warrants specific mention given that its translator is actively developed to minimise sensitivity to element ordering, with remaining cases tracked in issue 1158 in its repository; we cannot distinguish from our data whether FD's residual sensitivity originates in translator or other components. Practically, this heterogeneity means LLM-based configuration is better suited to scenarios where planner choice is flexible (e.g., portfolio planning) or where a specific planner consistently benefits; for planners that systematically degrade, it should be combined with planner-specific feedback.

Uses and Implications LLM-based configuration is not only a one-shot performance tool. Domain models evolve continuously through refactoring, extension, and cross-team integration and our approach can be embedded in this lifecycle as a diagnostic and exploratory tool. Generated variants serve as provenance artefacts (Georgievski 2023): each variant makes explicit a specific structural change and its observable effect on planner behaviour. Even variants that fail to improve performance carry diagnostic value, revealing which regions of a domain are sensitive under a given planner. Our single-forward-pass setting is a deliberate baseline rather than an endpoint; the natural next step is a closed-loop extension in which the LLM observes planner results and refines its outputs. Finally, the heterogeneous planner responses provide empirical evidence that different system architectures exhibit differential sensitivity to structural domain variations, motivating the investigation of which planner design choices relate to structural sensitivity.

Related Work

Our work bridges two lines of research: automated domain model configuration and the growing application of LLMs in planning. Vallati et al. (2015) introduced the first approach to domain model configuration by treating structural choices, such as the ordering of predicates, actions, and preconditions, as a configuration space and optimising them with a sequential model-based algorithm configuration method, relying on an offline learning phase in which many candidate configurations are evaluated on a set of training instances. Vallati et al. (2021) propose complementary online heuristics that focus primarily on action ordering. Georgievski et al. (2026) investigate the impact of domain model configuration on energy use of classical planners. More broadly, reformulation techniques have shown that structural transformations can substantially influence planner performance (Alarnaouti, Baryannis, and Vallati 2023). Related research focuses on domain model repair, which corrects flawed models rather than restructuring them and often requires significant expert input (Bercher, Sreetharan, and Vallati 2025).

Recent advances explore the use of LLMs for generating or refining PDDL models from natural language (Pallagani et al. 2024; Tantakoun, Muise, and Zhu 2025). Frameworks combining LLM generation with validation-guided feedback loops have shown moderate success in producing correct domains (Guan et al. 2023; Oswald et al. 2024; Goebel and Zips 2024; Smirnov et al. 2024; Huang and Zhang 2025; Casciani et al. 2025; Huang, Lipovetzky, and Cohn 2025). LLMs have also been used to generate domain-specific heuristics, action-choice policies, or generalised plans (Silver et al. 2024; Stein et al. 2025; Tuisov, Vernik, and Shleyfman 2025; Corrêa, Pereira, and Seipp 2025). Broader analyses emphasise that LLMs can support knowledge engineering in AI planning but require external validators and human oversight to ensure correctness (Vallati et al. 2025).

Conclusion

We present a framework for LLMs as domain model configurators, evaluated across seven LLMs, five prompting strategies, five planners, and five domains. Rather than framing the study as a performance-improvement exercise, we use it to characterise how LLMs restructure PDDL domain models and how planners respond. Three findings stand out. First, LLM choice is the primary determinant of configuration quality: capable models produce near-complete valid configurations with minimal prompting, while weaker models fail regardless of guidance; the simplest zero-shot prompt tends to outperform richer strategies, and temperature has no detectable effect. Second, LLMs do not confine themselves to the reordering that the task requires: roughly half of the candidates modify elements at the action, type, and requirement levels and are rejected by our validation, revealing a systematic bias in how LLMs interpret restructuring. Third, planner responses to the valid configurations are systematic but heterogeneous: SIW gains on all five domains, BFS_f degrades on four and shows the largest magnitude losses, and FD exhibits modest domain-dependent sensitivity. These findings motivate configuration-sensitivity stress tests in planner evaluation and suggest that LLM-based configuration is useful for diagnostics even when it does not improve performance. Natural extensions include deeper integration with planner architectures, fine-tuning on valid configurations, and refinement with planner feedback.

References

Alarnaouti, D.; Baryannis, G.; and Vallati, M. 2023. Reformulation Techniques for Automated Planning: A systematic Review. *The Knowledge Engineering Review*, 38: e9.

Anthropic. 2024. Claude 3.5 Sonnet. https://www.anthropic.com/news/claude-3-5-sonnet. Accessed: December 7, 2025.

Areces, C.; Bustos, F.; Dominguez, M.; and Hoffmann, J. 2014. Optimizing Planning Domains by Automatic Action Schema Splitting. In *International Conference on Automated Planning and Scheduling*, 11–19.

Bercher, P.; Sreetharan, S.; and Vallati, M. 2025. A Survey on Model Repair in AI Planning. In *International Joint Conference on Artificial Intelligence*, 10371–10380.

Casciani, A.; Giacomo, G. D.; Marrella, A.; and Weinhuber, C. 2025. A Requirements Engineering-Driven Methodology for Planning Domain Generation via LLMs with Invariant-Based Refinement. In *ICAPS 2025 Workshop LM4Plan*.

Cassano, F.; Li, L.; Sethi, A.; Shinn, N.; Brennan-Jones, A.; Ginesin, J.; Berman, E.; Chakhnashvili, G.; Lozhkov, A.; Anderson, C. J.; et al. 2023. Can It Edit? Evaluating the Ability of Large Language Models to Follow Code Editing Instructions. Preprint 2312.12450, arXiv.

Chrpa, L.; and Barták, R. 2009. Reformulating Planning Problems by Eliminating Unpromising Actions. In *Eighth Symposium on Abstraction, Reformulation, and Approximation*.

Chrpa, L.; Vallati, M.; and McCluskey, T. L. 2019. Inner entanglements: Narrowing the search in classical planning by problem reformulation. *Computational Intelligence*, 35(2): 395–429.

Corrêa, A. B.; Pereira, A. G.; and Seipp, J. 2025. Classical Planning with LLM-Generated Heuristics: Challenging the State of the Art with Python Code. In *Annual Conference on Neural Information Processing Systems*.

Dubey, A.; Jauhri, A.; Pandey, A.; et al. 2024. The Llama 3 Herd of Models. Preprint 2407.21783, arXiv.

Fox, M.; and Long, D. 2003. PDDL2. 1: An Extension to PDDL for Expressing Temporal Planning Domains. *Journal of Artificial Intelligence Research*, 20: 61–124.

Franco, S.; Vallati, M.; Lindsay, A.; and McCluskey, T. L. 2019. Improving planning performance in PDDL+ domains via automated predicate reformulation. In *International Conference on Computational Science*, 491–498.

Georgievski, I. 2023. Software Development Lifecycle for Engineering AI Planning Systems. In *International Conference on Software Technologies*, 751–760.

Georgievski, I.; Tekin, S.; and Aiello, M. 2026. The Energy Impact of Domain Model Design in Classical Planning. In *International Conference on AI Engineering – Software Engineering for AI*, 250–256. ACM/IEEE.

Goebel, K.; and Zips, P. 2024. Nl2plan: Robust LLM-Driven Planning from Minimal Text Descriptions. Preprint 2405.04215, arXiv.

Guan, L.; Valmeekam, K.; Sreedharan, S.; and Kambhampati, S. 2023. Leveraging pre-trained large language models to construct and utilize world models for model-based task planning. *Advances in Neural Information Processing Systems*, 36: 79081–79094.

Helmert, M. 2009. Concise finite-domain representations for PDDL planning tasks. *Artificial Intelligence*, 173(5-6): 503–535.

Howe, A. E.; and Dahlman, E. 2002. A Critical Assessment of Benchmark Comparison in Planning. *Journal of Artificial Intelligence Research*, 17: 1–33.

Huang, C.; and Zhang, L. 2025. On the Limit of Language Models as Planning Formalizers. In *Annual Meeting of the Association for Computational Linguistics*, 4880–4904.

Huang, S.; Lipovetzky, N.; and Cohn, T. 2025. Planning in the Dark: LLM-Symbolic Planning Pipeline Without Experts. In *AAAI Conference on Artificial Intelligence*, 26542–26550.

Jiang, A. Q.; Sablayrolles, A.; Roux, A.; et al. 2024. Mixtral of Experts. Preprint 2401.04088, arXiv.

Katz, M.; and Hoffmann, J. 2014. Mercury Planner: Pushing the Limits of Partial Delete Relaxation. In *Eight International Planning Competition: Planner Abstracts*, 43–47.

Kelly, J.; Calderwood, A.; Wardrip-Fruin, N.; and Mateas, M. 2023. There and Back Again: Extracting Formal Domains for Controllable Neurosymbolic Story Authoring. In *AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, volume 19, 64–74.

Kojima, T.; Gu, S. S.; Reid, M.; Matsuo, Y.; and Iwasawa, Y. 2023. Large Language Models are Zero-Shot Reasoners. Technical Report 2205.11916, arXiv.

Lipovetzky, N.; Ramirez, M.; Muise, C.; and Geffner, H. 2014. Width and Inference Based Planners: SIW, BFS(f), and PROBE. In *Eight International Planning Competition: Planner Abstracts*, 6–7.

Liu, X.; et al. 2024. DeepSeek-V2: A Strong, Economical, and Efficient Mixture-of-Experts Language Model. Preprint 2405.04434, arXiv.

Mahdavi, S.; Aoki, R.; Tang, K.; and Cao, Y. 2024. Leveraging environment interaction for automated PDDL translation and planning with large language models. In *International Conference on Neural Information Processing Systems*.

McCluskey, T. L.; Vaquero, T. S.; and Vallati, M. 2017. Engineering knowledge for automated planning: Towards a notion of quality. In *Proceedings of the 9th Knowledge Capture Conference*, 1–8.

McDermott, D.; Ghallab, M.; Howe, A.; Knoblock, C.; Ram, A.; Veloso, M.; Weld, D.; and Wilkins, D. 1998. PDDL - The planning domain definition language. Technical report, CVC TR-98-003/DCS TR-1165, Yale Center for Computational Vision and Control.

Minaee, S.; Mikolov, T.; Nikzad, N.; Chenaghlu, M.; Socher, R.; Amatriain, X.; and Gao, J. 2024. Large Language Models: A Survey. Preprint 2402.06196, arXiv.

OpenAI. 2024a. GPT-4o System Card. https://openai.com/index/gpt-4o-system-card/. Accessed: December 7, 2025.

OpenAI. 2024b. Learning to Reason with LLMs. https://openai.com/index/learning-to-reason-with-llms/. Accessed: December 7, 2025.

Oswald, J.; Srinivas, K.; Kokel, H.; Lee, J.; Katz, M.; and Sohrabi, S. 2024. Large language models as planning domain generators. In *International Conference on Automated Planning and Scheduling*, 423–431.

Pallagani, V.; Muppasani, B. C.; Roy, K.; Fabiano, F.; Loreggia, A.; Murugesan, K.; Srivastava, B.; Rossi, F.; Horesh, L.; and Sheth, A. 2024. On the Prospects of Incorporating Large Language Models (LLMs) in Automated Planning and Scheduling (APS). In *International Conference on Automated Planning and Scheduling*, volume 34, 432–444.

Reid, M.; Savinov, N.; Teplyashin, D.; et al. 2024. Gemini 1.5: Unlocking Multimodal Understanding Across Millions of Tokens of Context. Preprint 2403.05530, arXiv.

Riddle, P. J.; Holte, R. C.; and Barley, M. W. 2011. Does representation matter in the planning competition? In *Symposium of Abstraction, Reformulation, and Approximation*.

Rintanen, J. 2014. Madagascar: Scalable planning with SAT. In *Eight International Planning Competition: Planner Abstracts*, 1–5.

Röger, G.; Pommerening, F.; and Seipp, J. 2014. Fast downward stone soup 2014. In *Eight International Planning Competition: Planner Abstracts*, 28–31.

Silver, T.; Dan, S.; Srinivas, K.; Tenenbaum, J. B.; Kaelbling, L.; and Katz, M. 2024. Generalized Planning in PDDL Domains with Pretrained Large Language Models. In *AAAI conference on artificial intelligence*, 20256–20264.

Smirnov, P.; Joublin, F.; Ceravola, A.; and Gienger, M. 2024. Generating Consistent PDDL Domains with Large Language Models. Technical Report 2404.07751, arXiv.

Stein, K.; Fišer, D.; Hoffmann, J.; and Koller, A. 2025. Automating the Generation of Prompts for LLM-based Action Choice in PDDL Planning. In *International Conference on Automated Planning and Scheduling*, 250–259.

Tantakoun, M.; Muise, C.; and Zhu, X. 2025. LLMs as planning formalizers: A survey for leveraging large language models to construct automated planning models. In *Findings of the Association for Computational Linguistics: ACL 2025*, 25167–25188.

Tuisov, A.; Vernik, Y.; and Shleyfman, A. 2025. LLM-Generated Heuristics for AI Planning: Do We Even Need Domain-Independence Anymore? Technical Report 2501.18784, arXiv.

University of Potsdam. 2017. PDDL Benchmark Instance. https://github.com/potassco/pddl-instances. Accessed: December 7, 2025.

Vallati, M.; Barták, R.; Chrpa, L.; McCluskey, T. L.; and Petrick, R. P. 2025. Knowledge Engineering for Planning and Scheduling in the LLM Era. In *Proceedings of the International Conference on Automated Planning and Scheduling*, 391–395.

Vallati, M.; Chrpa, L.; McCluskey, T. L.; and Hutter, F. 2021. On the Importance of Domain Model Configuration for Automated Planning Engines. *Journal of Automated Reasoning*, 65(6): 727–773.

Vallati, M.; Chrpa, L.; and Serina, I. 2020. MEvo: a framework for effective macro sets evolution. *Journal of Experimental & Theoretical Artificial Intelligence*, 32(4): 685–703.

Vallati, M.; Hutter, F.; Chrpa, L.; and McCluskey, T. L. 2015. On the Effective Configuration of Planning Domain Models. In *International Joint Conference on Artificial Intelligence*, 1704–1711.

Valmeekam, K.; Olmo, A.; Sreedharan, S.; and Kambhampati, S. 2022. Large language models still can't plan (a benchmark for LLMs on planning and reasoning about change). In *NeurIPS 2022 Foundation Models for Decision Making Workshop*.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, L.; and Polosukhin, I. 2017. Attention is all you need. In *International Conference on Neural Information Processing Systems*, 6000–6010.

Wang, X.; Wei, J.; Schuurmans, D.; Le, Q.; Chi, E.; Narang, S.; Chowdhery, A.; and Zhou, D. 2022. Self-consistency improves chain of thought reasoning in language models. Technical Report 2203.11171, arXiv.

Zhao, W. X.; Zhou, K.; Li, J.; Tang, T.; Wang, X.; Hou, Y.; Min, Y.; Zhang, B.; Zhang, J.; Dong, Z.; et al. 2023. A Survey of Large Language Models. Preprint 2303.18223, arXiv.