Appendix of Early Product-Line Validation: Assessing LLMs for Analysis of Semi-Formal Blueprints

ACM Reference Format:
. 2026. Appendix of Early Product-Line Validation: Assessing LLMs for Analysis of Semi-Formal Blueprints. In *Proceedings of The 41st ACM/SIGAPP Symposium on Applied Computing (SAC'26).* ACM, New York, NY, USA, 8 pages. https://doi.org/XXXXXXX.XXXXXXX

A Task Instructions

This section details the reasoning procedures encoded in the user prompts for each analysis operation (AO).

Common preprocessing. For every AO, the model first normalizes the blueprint into a canonical representation: (i) enumerate all unique features and the root; (ii) parse relationship semantics (mandatory, optional, or, alternative); (iii) extract cross-tree constraints and rewrite them as logic (*requires*: $A \Rightarrow B$, *excludes*: $\neg(A \wedge B)$). This shared decomposition is referenced by all tasks below.

Solver-free structural metrics (AO1–AO9). Using only the canonical hierarchy and constraint list, the model: (1) counts unique features and leaf nodes; (2) computes maximum depth from the root; (3) tallies mandatory/optional features and numbers of or/alternative relationships; (4) counts requires/excludes constraints. All quantities are derived directly from the parsed structure, i.e., no external solver is invoked. When needed (e.g., for local checks), the model applies lightweight sanity rules that disambiguate relationship semantics before any counting.

Model satisfiability (AO10). Combine the hierarchical rules with all cross-tree constraints. Search for contradictions (e.g., mutually exclusive mandatory selections, cycles of implications that force an exclusion, illegal group assignments). If at least one configuration consistent with all rules can be constructed, output `true`; otherwise `false`. Provide a minimal conflicting subset when unsatisfiable.

Configuration satisfiability (AO11). This operation checks whether a given configuration of selected features is valid with respect to the rules of the feature model. The check proceeds in four steps: (1) verify that all mandatory features of selected parents are included, and that feature groups (or, alternative) are chosen with valid multiplicities (e.g., exactly one for alternative, at least one for or); (2) ensure that all requires constraints are satisfied, including indirect ones (if A requires B and B requires C, selecting A must also include C); (3) verify that no pair of excludes features is selected together; and (4) confirm that no additional features implied by these constraints cause group violations (for example, if two required features belong to the same alternative group). If all

Author's Contact Information:

SAC'26, Thessaloniki, Greece

ACM ISBN 979-X-XXXX-XXXX-X/26/03
https://doi.org/XXXXXXX.XXXXXXX

four checks hold, the configuration is considered satisfiable; otherwise, the smallest conflicting subset of features or constraints is reported.

Number of valid configurations (AO12). Perform a constraint-aware branch-and-count over the feature tree: (1) factorize by relationships (mandatory, optional, or, alternative) to form local choice sets; (2) prune branches early using requires/excludes constraints; (3) count only branches that satisfy all constraints. Where applicable, apply small-scale inclusion–exclusion on independent subtrees as a cross-check.

Core features (AO13). For each feature f, attempt a witness *without* f while satisfying all constraints. If no valid configuration omitting f exists, classify f as *core*. Justify with the blocking dependency (mandatory lineage, chained requires, or group logic).

Dead features (AO14). For each feature f, attempt a witness *with* f selected that respects all relationships and constraints. If every attempt yields a contradiction (e.g., triggers an excludes or violates group multiplicity), mark f as *dead* and state the tightest conflicting set.

False optional features (AO15). Start from features that appear optional (explicitly optional or group members that could be omitted). For each candidate f, try to build a valid configuration where its parent is selected but f is not. If no such configuration exists, f is *false optional*. Explain which dependency or constraint forces f.

Generalization (AO16). Given an original blueprint M and its edited variant M', judge whether M' preserves all configurations of M. Attempt to find a counterexample: a configuration valid in M but invalid in M'. If none is found under the parsed semantics, conclude that M' generalizes M; otherwise report the counterexample and the violated rule in M'.

B Further Results

B.1 Accuracy of LLM-based AOs

Table 1. Accuracy (%) of general-purpose LLMs across 9 solver-free AOs.

Model ID	AO1	AO2	AO3	AO4	AO5	AO6	AO7	AO8	AO9	Average
grok-4-non-reasoning	40	40	30	20	50	90	80	60	80	54.4
gpt-4.1	50	40	50	30	50	100	70	70	80	60.0
llama-4-scout	30	20	30	20	30	40	50	40	70	36.7
claude-sonnet-4	100	50	80	40	70	100	70	80	90	75.6
deepseek-chat	30	30	60	10	40	100	70	60	80	53.3
Average	50.0	36.0	50.0	24.0	48.0	86.0	68.0	62.0	80.0	56.0

Table 2. Accuracy (%) of reasoning-optimized LLMs across 9 solver-free AOs.

Model ID	**AO1**	**AO2**	**AO3**	**AO4**	**AO5**	**AO6**	**AO7**	**AO8**	**AO9**	**Average**
grok-4-reasoning	100	80	100	80	90	100	80	90	100	91.1
gemini-2.5-flash	60	60	80	50	80	80	80	70	80	71.1
gemini-2.5-pro	100	70	90	80	80	100	80	90	100	87.8
llama-4-maverick	60	30	30	30	60	80	70	60	80	55.6
gpt-5-mini	100	80	70	80	80	100	80	100	100	87.8
claude-sonnet-4-think	100	70	80	60	80	100	70	80	90	81.1
deepseek-reasoner	90	80	100	30	50	100	80	90	100	80.0
Average	87.1	67.1	78.6	58.6	74.3	94.3	77.1	82.9	92.9	79.2

Table 3. Accuracy (%) of general-purpose LLMs on 9 solver-free AOs across blueprints.

Model ID	SW	SMW	IDE	SMG	COM	SEA	CVE	BDB	CNNl	CNNf
grok-4-non-reasoning	100	89	78	78	44	56	56	33	11	0
gpt-4.1	100	100	67	67	33	56	56	78	22	22
llama-4-scout	89	78	56	44	22	11	22	33	11	0
claude-sonnet-4	100	100	89	89	100	56	78	56	33	56
deepseek-chat	89	78	56	60	22	44	78	44	11	44
Average	95.6	89.0	69.2	67.6	44.2	44.6	58.0	48.8	17.6	24.4

Table 4. Accuracy (%) of reasoning-optimized LLMs on 9 solver-free AOs across blueprints.

Model ID	SW	SMW	IDE	SMG	COM	SEA	CVE	BDB	CNNl	CNNf
grok-4-reasoning	100	100	100	100	89	100	100	100	67	56
gemini-2.5-flash	100	100	100	89	100	67	67	89	0	0
gemini-2.5-pro	100	100	100	100	100	89	100	100	44	44
llama-4-maverick	100	100	78	78	44	67	44	44	0	0
gpt-5-mini	100	100	100	100	100	100	100	89	44	44
claude-sonnet-4-think	100	100	100	100	100	67	89	89	22	44
deepseek-reasoner	100	89	78	78	100	89	78	78	56	56
Average	100.0	98.4	93.7	92.1	90.4	82.7	82.6	84.1	33.3	34.9

Table 5. Accuracy (%) of general-purpose LLMs across solver-based AOs.

Model ID	AO10	AO11	AO12	AO13	AO14	AO15	AO16	Average
grok-4-non-reasoning	90	80	50	50	70	60	100	71.4
gpt-4.1	100	100	50	80	70	60	100	80.0
llama-4-scout	90	90	13	40	40	20	40	47.6
claude-sonnet-4	100	90	50	90	80	70	90	81.4
deepseek-chat	80	80	25	60	60	50	90	63.6
Average	92.0	88.0	37.6	64.0	64.0	52.0	84.0	68.8

Table 6. Accuracy (%) of reasoning-optimized LLMs across solver-based AOs.

Model ID	AO10	AO11	AO12	AO13	AO14	AO15	AO16	Average
grok-4-reasoning	100	100	100	60	90	70	100	88.6
gemini-2.5-flash	100	90	63	70	80	60	80	77.6
gemini-2.5-pro	100	90	100	90	90	80	100	92.9
llama-4-maverick	70	100	13	60	60	40	90	61.9
gpt-5-mini	100	100	75	90	90	80	100	90.7
claude-sonnet-4-think	100	100	75	90	90	80	80	87.9
deepseek-reasoner	100	90	75	90	80	70	90	85.0
Average	95.7	95.7	71.6	78.6	82.9	68.6	91.4	83.5

Table 7. Accuracy (%) of general-purpose LLMs on 7 solver-based AOs across blueprints.

Model ID	SW	SMW	IDE	SMG	COM	SEA	CVE	BDB	CNNl	CNNf
grok-4-non-reasoning	100	100	86	86	71	43	43	57	67	67
gpt-4.1	100	100	100	100	86	71	43	57	50	100
llama-4-scout	57	100	43	57	43	43	71	43	17	33
claude-sonnet-4	100	100	100	86	86	71	100	57	50	83
deepseek-chat	100	86	86	71	71	57	57	43	33	83
Average	91.4	97.2	83.0	80.0	71.4	57.0	62.8	51.4	43.4	73.2

Table 8. Accuracy (%) of reasoning-optimized LLMs on 7 solver-based AOs across blueprints.

Model ID	SW	SMW	IDE	SMG	COM	SEA	CVE	BDB	CNNl	CNNf
grok-4-reasoning	100	100	100	100	100	86	71	71	50	100
gemini-2.5-flash	100	100	100	86	86	57	57	57	67	67
gemini-2.5-pro	100	100	100	100	100	71	100	57	67	100
llama-4-maverick	100	86	86	86	57	43	43	29	33	67
gpt-5-mini	100	100	100	100	100	100	86	57	67	100
claude-sonnet-4-think	100	100	100	86	100	71	100	71	67	100
deepseek-reasoner	100	100	86	100	86	86	86	71	67	83
Average	100.0	98.0	96.0	94.0	90.0	73.4	77.6	59.0	59.7	88.1

Table 9. Accuracy (%) of general-purpose LLMs on 16 AOs across blueprints.

Model ID	SW	SMW	IDE	SMG	COM	SEA	CVE	BDB	CNNl	CNNf	**Average**
grok-4-non-reasoning	100	94	81	81	56	50	50	44	31	25	61.2
gpt-4.1	100	100	81	81	56	62	50	69	31	50	**68.0**
llama-4-scout	75	88	50	50	31	25	44	38	13	13	42.7
claude-sonnet-4	100	100	94	88	94	62	88	56	38	63	**78.3**
deepseek-chat	94	81	69	62	44	50	69	44	19	56	59.9
Average	93.8	92.6	75.0	72.4	56.2	49.8	60.2	50.2	26.4	41.4	61.0

Note. Bold numbers indicate models achieving the best overall performance; underlined values denote challenging feature models where accuracy is low.

Table 10. Accuracy (%) of reasoning-optimized LLMs on 16 AOs across blueprints.

Model ID	SW	SMW	IDE	SMG	COM	SEA	CVE	BDB	CNNl	CNNf	**Average**
grok-4-reasoning	100	100	100	100	94	94	88	88	60	73	**89.7**
gemini-2.5-flash	100	100	100	88	94	62	62	75	27	27	73.5
gemini-2.5-pro	100	100	100	100	100	81	100	81	53	67	**88.2**
llama-4-maverick	100	94	81	81	50	56	44	38	13	27	62.4
gpt-5-mini	100	100	100	100	100	100	94	75	53	67	**88.9**
claude-sonnet-4-think	100	100	100	94	100	69	94	81	40	67	84.5
deepseek-reasoner	100	94	81	88	94	88	81	75	60	67	82.8
Average	100.0	98.3	94.6	93.0	90.3	78.6	80.4	74.7	43.7	56.3	81.1

Note. Bold numbers indicate models achieving the best overall performance; underlined values denote challenging feature models where accuracy is low.

B.2 Cost of LLM-based AOs

Table 11. Average runtimes (in seconds) of each LLM for 16 AOs on 10 blueprints, alongside FLAMA's performance.

Model ID	SW	SMW	IDE	SMG	COM	SEA	CVE	BDB	CNNl	CNNf	Average
grok-4-non-reasoning	101	249	403	610	881	1217	1334	1636	1987	2275	1169.3
gpt-4.1	133	323	542	881	1374	2000	2285	2783	4128	4580	1892.9
llama-4-scout	88	196	294	382	492	683	751	881	1287	1748	680.2
claude-sonnet-4	124	269	443	686	895	1199	1375	1660	1871	2063	1168.5
deepseek-chat	197	453	752	1082	1572	2063	2330	2811	3344	3891	1849.5
grok-4-reasoning	221	535	912	1214	1701	2753	3050	4152	5294	5835	3046.7
gemini-2.5-flash	124	286	471	753	1342	2382	2654	3750	5764	7699	3222.5
gemini-2.5-pro	168	366	586	850	1256	1861	2155	2963	3952	4951	2390.8
llama-4-maverick	67	156	279	449	653	839	902	976	1472	2068	886.1
gpt-5-mini	479	1122	1982	3245	4827	6730	7769	10567	12017	13045	6118.3
claude-sonnet-4-think	270	592	971	1403	2043	3172	3573	4715	5489	5983	3011.1
deepseek-reasoner	815	1948	3428	5263	7617	10603	12029	15665	18181	20235	9508.4
FLAMA (Solver)†	0.0005	0.0011	0.0015	0.1037	0.3522	60.0982	0.1215	0.0372	60.3814	61.4404	18.45

†Results on formal inputs; solver cannot process semi-formal blueprints.

Table 12. Token consumptions of each LLM for 16 AOs on 10 blueprints.

Model ID	SW	SMW	IDE	SMG	COM	SEA	CVE	BDB	CNNl	CNNf	Average
grok-4-non-reasoning	24169	51050	80823	115685	160722	217493	267548	344634	701598	1412267	397599.0
gpt-4.1	23733	49499	76478	106991	147857	198643	247339	316405	662554	1365923	379042.2
llama-4-scout	24168	49765	75602	104333	137007	180042	225816	285524	647489	1368337	374608.3
claude-sonnet-4	28149	58272	91645	126395	168487	221851	274387	344894	707425	1287725	439523.0
deepseek-chat	23629	49720	77330	107658	145780	190342	237267	301916	649363	1206488	377549.3
grok-4-reasoning	29041	63488	101618	148517	213478	301704	364204	473289	902899	1623506	505874.4
gemini-2.5-flash	38375	84453	136414	204045	305885	504374	595520	828668	1639393	2882763	765789.0
gemini-2.5-pro	40145	84391	132640	188385	268049	383351	470076	635049	1210436	2260989	602251.1
llama-4-maverick	24363	52119	81078	113823	150870	193929	242472	302917	678755	1412195	389052.1
gpt-5-mini	49531	111724	182992	282924	414198	578517	693933	916616	1345818	2107258	843251.1
claude-sonnet-4-think	37170	81463	129360	180836	249601	354763	422937	538218	940241	1539850	551644.0
deepseek-reasoner	39384	88434	147229	216841	304018	414756	492170	635532	1036354	1637181	561290.0

B.3 Error Analysis

Taxonomy and Overall Frequencies. We categorize failures into five mutually exclusive types: (i) *Format-compliant but wrong* (syntactically valid, semantically incorrect), (ii) *Incomplete / missing answers* (typically due to context/output limits), (iii) *Nonsense text* (unparsable natural language despite a valid contract elsewhere), (iv) *Contract violations* (malformed or missing required tags), and (v) *Refusals* (model explicitly declines to compute, e.g., "a solver is required"). Table 14 summarizes counts aggregated over all runs.

Where and why models fail. (1) **Semantic misinterpretation (structural AOs).** The dominant source of error in solver-free AOs (A04–A09) is misunderstanding of group semantics, especially reading "*A must have B or C*" as two mandatory children instead of an alternative/or group. This yields inflated #mandatory and deflated #alternative counts and cascades into wrong #requires/#excludes. The effect is most visible on medium/large blueprints (SMG, SEA, BDB, CNNl, CNNf).

(2) **Propagation/enumeration limits (reasoning AOs).** A012 (#valid configurations) and A015 (#false optional) require either partial enumeration or precise constraint propagation. Many models under-approximate constraints (missed implications) or over-approximate (treating exclusive groups as independent), producing wrong-but-formatted outputs. A013/A014 (core/dead features) show similar patterns when cross-tree constraints are dense (BDB, CVE).

Table 13. Average runtimes (in seconds) and token consumptions of each LLM across 16 AOs on 10 blueprints.

Model ID	Avg. Runtime	Avg. Tokens
grok-4-non-reasoning	1,169.3	397,599
gpt-4.1	1,892.9	379,042
llama-4-scout	680.2	374,608
claude-sonnet-4	1,168.5	439,523
deepseek-chat	1,849.5	377,549
grok-4-reasoning	3,046.7	505,874
gemini-2.5-flash	3,222.5	765,789
gemini-2.5-pro	2,390.8	602,251
llama-4-maverick	886.1	389,052
gpt-5-mini	6,118.3	843,251
claude-sonnet-4-think	3,011.1	551,644
deepseek-reasoner	9,508.4	561,290
FLAMA (Solver)[†]	18.45	–

[†]Results on formal inputs; solver cannot process semi-formal blueprints.

Table 14. Failure counts per model (all blueprints & AOs). The four rightmost error types appear almost exclusively on `CNNl` and `CNNf`.

Model	Wrong	Incomplete	Nonsense	Contract	Refusal
grok-4-non-reasoning	48	1	0	0	0
gpt-4.1	49	0	0	0	0
llama-4-scout	75	6	5	**2**	0
claude-sonnet-4	33	1	0	0	0
deepseek-chat	62	3	0	0	0
grok-4-reasoning	17	0	0	0	0
gemini-2.5-flash	21	**27**	0	0	0
gemini-2.5-pro	18	0	0	0	0
llama-4-maverick	51	6	5	**1**	0
gpt-5-mini	**7**	0	0	0	**9**
claude-sonnet-4-think	23	1	0	0	0
deepseek-reasoner	27	1	0	0	0

(3) **Context/output constraints (very large blueprints).** On `CNNl` and `CNNf`, four failure modes spike: *Incomplete*, *Nonsense*, *Contract*, and *Refusal*. Llama 4 Maverick (16K max output) frequently truncates and resorts to summaries (e.g., reporting only "329 features" for `CNNl`), leading to *Incomplete*/*Nonsense* and occasional *Contract* violations. Gemini 2.5 Flash exhibits many *Incomplete* runs (early stopping near context limits). GPT-5 mini issues *Refusals* (9×) stating a solver is required for exact counts on the largest instances. By contrast, Grok 4 Fast Reasoning avoids these failure modes but still accumulates wrong-but-formatted errors on the most complex AOs.

Model-specific patterns. Llama 4 Scout and DeepSeek Chat accumulate many wrong-but-formatted outputs on structural AOs; Scout also shows *Nonsense* and *Contract* errors on `CNNl/f`. Claude Sonnet

4 achieves strong results on small blueprints but consistently over-counts #mandatory when parsing alternative/or groups (IDE/SMG/SEA/CVE), and under-performs on #valid configurations for COM, SEA, BDB. Grok 4 Fast Reasoning, GPT-5 mini, and Gemini 2.5 Pro are markedly more stable; residual errors concentrate on AO12/AO15 and structural counts on CNNl/CNNf. DeepSeek Reasoner spends substantial runtime but still accumulates wrong-but-formatted errors on structural AOs (mandatory/optional) and AO15. Gemini 2.5 Flash fails open under long contexts (*Incomplete*). Llama 4 Maverick is dominated by *Incomplete*/*Nonsense* due to output truncation.

Blueprint- and AO-level hotspots. Failure rates rise with blueprint complexity: SEA (depth 10) and BDB (dense cross-tree constraints) trigger propagation mistakes; CNNl/CNNf trigger context/output issues and semantic slips in group interpretation. Across AOs, the hardest are AO12 and AO15 (propagation/enumeration); next are AO4–AO9 (semantic parsing); the easiest are verification-style AO10/AO11/AO16.

Illustrative failure modes. Semantic slip: "*A must have B or C*" → counted as two mandatory (inflated #mandatory, deflated #alternative). *Partial propagation*: treating an exclude as local (ignoring transitive implications) → under-count of dead/core features. *Output truncation*: reasoning stops mid-list (missing features/relationships) → *Incomplete* with plausible but wrong totals. *Refusal*: explicit claim that exact counting *requires a solver* (observed in GPT-5-Mini on CNNl/CNNf).

Summary. Most errors arise from (i) semantic misinterpretation in structural AOs and (ii) incomplete propagation/enumeration in reasoning AOs; very large blueprints additionally expose (iii) context/output limits. Reasoning-optimized models reduce but do not eliminate these failures.