Business as *Rule*sual: A Benchmark and Framework for Business Rule Flow Modeling with LLMs

Anonymous ACL submission

Abstract

Extracting structured procedural knowledge from unstructured business documents is a critical yet unresolved bottleneck in process automation. While prior work has focused on extracting linear action flows from instructional texts (e.g., recipes), insufficiently addressing the complex logical structures—such as conditional branching and parallel execution—that are pervasive in real-world regulatory and administrative documents. Furthermore, existing benchmarks are further limited by simplistic schemas and shallow logical dependencies, restricting progress toward logic-aware large language models (LLMs). To bridge this "Logic Gap", we introduce **BREX**, a carefully curated benchmark comprising 409 real-world business documents and 2,855 expert-annotated rules. Unlike prior datasets centered on narrow service scenarios, BREX spans over 30 vertical domains, covering scientific, industrial, administrative, and financial regulations.

We further propose **ExIde**, a structure-aware reasoning framework that investigates five distinct prompting strategies, ranging from implicit semantic alignment to executable grounding via pseudo-code generation, enabling explicit modeling of rule dependencies and providing out-of-the-box framework to different business customers without finetuning their own LLMs. We benchmark ExIde using 13 state-of-the-art LLMs. Our extensive evaluation reveals that: (1) Executable grounding serves as a superior inductive bias, significantly outperforming standard prompts in rule extraction; and (2) Reasoning-optimized models demonstrate a distinct advantage in tracing long-range dependencies and non-linear rule dependencies compared to standard instruction-tuned models.

1 Introduction

Modern enterprises rely on extensive collections of natural language regulations to govern complex operational processes, ranging from safety protocols in physics laboratories to compliance checks in financial services (Kourani et al., 2024; Zhang et al., 2025). Although these documents are written for human interpretation, their execution in real-world systems requires structured, machine-readable representations that explicitly encode logical constraints and dependencies. In practice, this translation is still performed manually by domain experts using rule engines or workflow languages (e.g., executable code or BPMN models (Wohed et al., 2006)), making the process labor-intensive, error-prone, and difficult to maintain under frequent policy updates (Sivasankari et al., 2020).

We study a fundamental yet underexplored problem: *how can unstructured business manuals be automatically transformed into structured rule flows that explicitly capture conditional branching, parallel constraints, and inter-rule dependencies?* We refer to this challenge as **Logic Gap**: the discrepancy between free-form natural language regulations and the executable, condition-dependent control flow required by automated systems.

Despite its practical importance, the Logic Gap remains poorly addressed in existing NLP benchmarks. Prior work on procedural text understanding (Quishpi et al., 2020; Du et al., 2024; Redis et al., 2024; Pyrih et al., 2025) predominantly focuses on extracting *action-centric* process flows, often modeling procedures as linear or weakly structured event sequences (e.g., recipes or tutorials). Such representations are insufficient for professional regulatory documents, where actions are governed by nested conditions, branching decisions, and parallel requirements. For example, in financial compliance, a rule such as *"If the currency type is USD, select remittance type"* is activated by a condition rather than a preceding action, and subsequent checks may be triggered conditionally or executed in parallel.

This distinction is critical: *action flows* describe

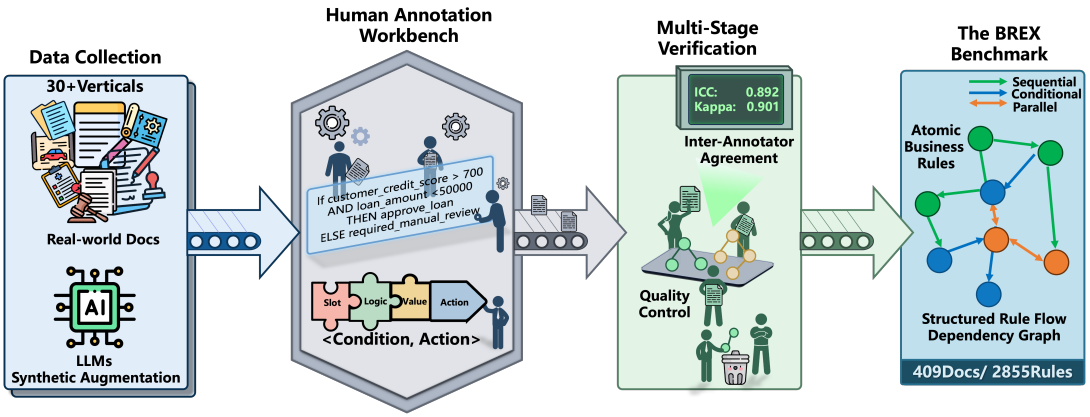

Figure 1: Construction pipeline of the BREX benchmark: (1) **Data collection**, combining real-world business documents with a limited amount of expert-filtered synthetic data; (2) **Expert annotation**, where three domain experts annotate atomic rules and their dependencies; and (3) **Multi-stage verification**, including quality control with another three experts and inter-annotator agreement evaluation using ICC and Kappa.

what actions occur, whereas *rule flows* specify the logical conditions under which actions are permitted or constrained. However, rule-centric modeling has been largely overlooked due to the lack of benchmarks that explicitly annotate both atomic business rules and their inter-rule dependencies at scale. As a result, existing datasets provide limited support for developing and evaluating logic-aware LLMs.

To fill this gap, we introduce **BREX** (Business Rule Extraction Benchmark), a dataset specifically designed for business rule flow modeling (Fig. 1). BREX consists of 409 real-world business documents and 2,855 expert-annotated rules spanning over 30 vertical domains, including scientific research management, industrial manufacturing, administrative approvals, and financial compliance. Each rule is formalized as a structured condition–action pair and linked via explicit dependency relations (*Sequential*, *Conditional*, and *Parallel*), enabling systematic evaluation of both local rule extraction and global logical reasoning.

While many prior information extraction approaches rely on supervised fine-tuning, real-world deployment faces significant barriers, including strict data privacy constraints, high computational costs, and the challenge of adapting to frequent policy updates across heterogeneous formats. As a result, there is a strong industrial demand for **out-of-the-box solutions** that can operate directly on general-purpose LLMs without task-specific retraining. Motivated by this, we propose **ExIde** designed to maximize the potential of prompt-based inference. Rather than treating rule extraction as a flat information extraction task, ExIde adopts a *decompose-and-reason* strategy that introduces intermediate representations as inductive signals. We investigate five prompting strategies, ranging from implicit semantic alignment to executable grounding via pseudo-code generation, and benchmark them across 13 state-of-the-art LLMs. Our experiments reveal two key findings. First, executable grounding provides a strong inductive guidance for recovering complex business rules, consistently outperforming standard prompting strategies. Second, reasoning-optimized models exhibit a clear advantage in tracing long-range and non-linear rule dependencies, highlighting the importance of structured reasoning for logic-intensive extraction tasks [1].

In summary, our contributions are threefold:

- We introduce BREX, a cross-domain benchmark for business rule flow modeling with expert-annotated logical conditions and dependencies across 30+ real-world domains.
- We propose ExIde, a structure-aware framework that leverages executable grounding to bridge natural language regulations and machine-executable rule flows, and provide an out-of-the-box solutions to diverse business customers without finetuning their own LLMs.
- We provide a comprehensive empirical study of 13 LLMs, offering new insights into the role of executable representations and reasoning-oriented model design for logic-intensive extraction tasks.

[1] We will release our annotated datasets after the paper acceptance.

2 Related Work

2.1 From Action-Centric to Rule-Centric Modeling

Prior procedural text understanding predominantly focuses on *action-centric* modeling. Early studies applied extraction techniques to instructional texts (Maeta et al., 2015; Friedrich et al., 2011), while later work extended this to structured documents using syntactic or neural models (Guo et al., 2018; Epure et al., 2015; Ren et al., 2023; Pal et al., 2021; Candido et al., 2024). However, these approaches typically conceptualize processes as linear action sequences derived from temporal order. They struggle with real-world regulations where actions are governed by complex *interdependent business rules* involving conditional branching and parallel execution. This limitation is exacerbated by the lack of benchmarks explicitly annotating rule-to-rule dependencies. A detailed comparison highlighting these differences in modeling paradigms and data sources is provided in Table 6 in Appendix A.4.

2.2 LLMs for Process Understanding

Recent work has investigated LLMs for process modeling but remains limited in logical depth. While early studies extracted linear control flows (Bellan et al., 2022a), large-scale benchmarks like PAGED (Du et al., 2024) rely on synthetic data-to-text generation, often lacking the linguistic ambiguity and logical complexity of authentic regulatory documents. Although some approaches integrate formal constraints or instruction tuning (Pyrih et al., 2025), they typically operate under predefined schemas or focus on action-level semantics rather than atomic business rules. In contrast, we focus on *business rule flow modeling*, shifting from action sequences to jointly extracting atomic condition–action rules and reconstructing their global dependency structures. This enables a rigorous evaluation of LLMs' ability to capture the non-linear logic pervasive in real-world enterprises.

3 BREX Dataset

We introduce **BREX**, a benchmark designed for *business rule flow modeling*, which requires jointly extracting atomic business rules and recovering their logical dependencies from unstructured text. Given a business document, the task is to identify a set of condition–action rules and reconstruct a dependency graph.

3.1 Task Formalization and Annotation Schema

Each business rule is formalized as an atomic condition–action pair. The condition is represented as a structured triple ⟨Slot Type, Logical Operator, Reference Value⟩, while the action specifies the operation triggered when the condition is satisfied. Formal definitions are provided in Appendix A.1, and an annotated example is illustrated in Table 1. Complex rules involving multiple constraints are normalized into atomic units and linked through explicit dependency relations, enabling compositional modeling of non-linear logic.

We consider three types of dependency relationships: *Sequential*, where one rule must be executed before another; *Conditional*, where different outcomes of a rule trigger different subsequent rules; and *Parallel*, where multiple rules must be executed concurrently. Illustrative examples of these dependencies are provided in Appendix A.2.

3.2 Dataset Construction

To ensure both coverage and realism, we combine publicly available real-world documents with a limited amount of synthetic data generated using Gemini 2.5 Pro (Comanici et al., 2025). Importantly, synthetic texts are used only to augment underrepresented logical structures (e.g., nested conditions and parallel constraints) and are strictly filtered and revised by domain experts to ensure consistency with real-world regulatory language. The construction process (Fig 1) involves:

1. **Data Collection**: We gathered business texts from over **30 distinct vertical domains**. Crucially, unlike prior datasets limited to simple service interactions, BREX covers: Scientific & Industrial Logic, Administrative & Legal Logic, Service & Transactional Logic, etc,. Synthetic texts were generated to augment these specific domains. The generation prompt is shown in Fig 9 in Appendix.
2. **Expert Annotation**: Three domain experts annotated atomic rules and their dependency relations following a unified schema.
3. **Multi-Stage Verification**: To ensure the quality and consistency of the annotations, a separate team of three experts reviewed all annotations. Any ambiguous rules were discarded or re-annotated to ensure the dataset serves as a gold-standard benchmark.

The BREX dataset contains **409 documents** and

Our bank supports up to 39 currency types of popular countries or regions around the world, including RMB, USD, JPY, GBP, and HKD. After selecting the appropriate currency, the customer needs to choose the corresponding cash or remittance type based on the type of business to be conducted thereafter. The cash or remittance type includes cash and remittance. Upon completing the cash/remittance type selection, the customer needs to provide the purchase amount...			
Slot Type	**Logical Operator**	**Reference Value**	**Action**
currency types	including	RMB, USD, JPY, GBP, and HKD.	choose the corresponding cash or remittance type
The cash or remittance type	includes	cash and remittance	the customer needs to provide the purchase amount

Table 1: An example of business rule annotation. The model is required to extract each condition and action as a structured tuple $\langle\langle$Slot Type, Logical Operator, Reference Value$\rangle$, Action$\rangle$ and infer the corresponding rule dependencies. **Colors** denote corresponding components in the text.

2,855 business rules, with an average of **7 rules per document**. See Table 5 in Appendix for more statistics. Notably, while sequential dependencies are common, over 30% of rules participate in conditional or parallel relations, highlighting the prevalence of non-linear logic in real-world regulations.

3.3 Dataset Analysis

We assessed the quality of BREX from two perspectives: the linguistic quality of the texts and the reliability of the logical annotations.

Text Quality Assessment: Following prior studies (Miller, 1979; Du et al., 2024), three annotators rated the business texts on Readability, Accuracy, Clarity, Simplicity, and Usability (Scale 1-5) (See Appendix A.5). As shown in Fig 8 in Appendix, the texts received high ratings across all dimensions. We computed the Intraclass Correlation Coefficient (ICC) (Shrout and Fleiss, 1979) to measure agreement. The average ICC was **0.892**, indicating excellent consistency and high-quality textual data.

Inter-Annotator Agreement (IAA): Evaluating IAA for structured rule extraction is challenging as it involves both span identification and relation classification. To our knowledge, there is no established metric for directly measuring inter-annotator agreement on structured rule dependency graphs. To address this, we adopted a proxy evaluation method by projecting the rule annotations into a Named Entity Recognition (NER) format (as shown in Table 7 in A.6). Three NLP experts annotated a subset of data using BIO tagging for Slot Types, Reference Values and Action. We then computed Fleiss' Kappa (Artstein, 2017). The resulting score of **0.901** indicates *almost perfect agreement*, confirming that our definitions of business rules are unambiguous and the annotations are highly reliable.

4 Methodology

We propose **ExIde**, a structure-aware framework for *business rule flow modeling*. Given a business document x, ExIde aims to recover (i) a set of atomic rules $R = \{r_i\}$ and (ii) a dependency graph G that encodes *Sequential*, *Conditional*, and *Parallel* relations among rules. As illustrated in Fig. 2, ExIde adopts a *decompose-and-reason* strategy that separates rule extraction from global dependency reasoning.

4.1 Stage I: Structure-Aware Rule Extraction

In the first stage, ExIde extracts atomic rules of the form $r_i = (c_i, a_i)$, where each condition $c_i = \langle s_i, j_i, v_i \rangle$ follows the schema defined in Section 3, and a_i denotes the corresponding action. Rather than treating rule extraction as a flat information extraction task, ExIde introduces *intermediate reasoning structures* to guide LLMs toward logic-consistent outputs.

To study the effect of different inductive biases, we design five prompting strategies (P1–P5) that share the same output schema but differ in how intermediate reasoning is encouraged. These strategies range from implicit semantic alignment to explicit executable grounding. Specifically, Prompt 5 introduces an intermediate pseudo-code representation, in which the business logic is first expressed using simple procedural primitives (e.g., conditional branches) before structured rules are extracted. This design encourages early resolution of nested conditions and control flow, providing a structural scaffold for logic-intensive extraction.

All prompting strategies adopt a chain-of-thought style reasoning format (Wei et al., 2022) (see Fig. 10), but only the final structured outputs are used for evaluation. The specific reasoning

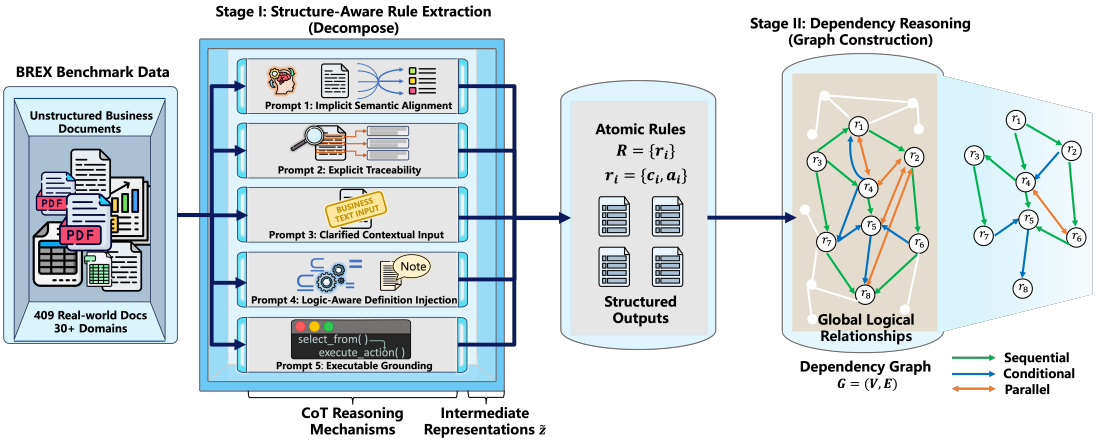

Figure 2: Overview of the **ExIde** framework. It adopts a *decompose-and-reason* strategy: (1) **Structure-Aware Extraction**, which employs five distinct reasoning mechanisms (e.g., executable grounding) to extract atomic rules; and (2) **Dependency Reasoning**, which infers global logical relationships (Sequential, Conditional, Parallel) among the extracted rules.

mechanisms are as follows:

- **Prompt 1: Implicit Semantic Alignment.** This prompt (Fig 11) establishes a baseline reasoning path. It requires the LLMs to first generate a natural language explanation (**Explain** section) that summarizes the business logic before populating the structured **Output**. The mapping between the source text and the extracted rules is implicit, relying on the model's internal attention mechanism to align semantics with the schema.
- **Prompt 2: Explicit Traceability (Alignment).** To mitigate hallucinations and enhance interpretability, this strategy (Fig 12) enforces a strict one-to-one mapping. The model must explicitly link each explanatory sentence to its corresponding business rule using declarative phrasing (e.g., "*This sentence corresponds to the business rule: ...*"). By forcing explicit alignment, we aim to reduce the generation of unsupported rules.
- **Prompt 3: Clarified Contextual Input.** Building on Prompt 1, this variant refines the input by explicitly labeling the "**Input**" field as "**Business Text Input**". While a subtle modification, this aims to prime the model's domain awareness, ensuring the input is processed strictly as regulatory text rather than general prose.
- **Prompt 4: Logic-Aware Definition Injection.** Complex logical operators (e.g., distinguishing *contains* vs. *equal to*) are frequent sources of error. This prompt (Fig 13) injects detailed constraints into the context, specifically guiding the handling of multi-value slot types. It introduces a dedicated "Note" section to disambiguate set inclusion from equality, enforcing rigorous adherence to the logical definitions in Section A.1.
- **Prompt 5: Executable Grounding (Pseudo-Code).** Prompt 5 (Fig 14) introduces an intermediate pseudo-code representation $\tilde{z}$ before producing the final rules. The model first translates x into pseudo-code using a small set of primitives (e.g., *select_from()*, *execute_action()*), and then extracts R from $\tilde{z}$. This design encourages early resolution of nested conditions and control-flow, leveraging code-like structure as an inductive bias for logic-intensive extraction.

Prompts 3, 4, and 5 represent targeted modifications to the baseline (Prompt 1), allowing us to systematically evaluate the impact of context clarification, logic definition, and code grounding on extraction performance.

4.2 Stage II: Dependency Graph Reconstruction

In the second stage, ExIde reconstructs a global dependency graph $G = (V, E)$ over the extracted rules. Each node $v_i \in V$ corresponds to a rule r_i, and each directed edge $e_{ij} \in E$ is labeled as *Sequential*, *Conditional*, *Parallel*, or *None*. Dependency identification is formulated as a pairwise classification problem over rule pairs. We employ a dedicated dependency reasoning prompt (Fig. 15) that takes rule pairs as input and predicts their dependency type. The resulting pairwise predictions are assembled into a typed adjacency matrix, which is then used to construct a coherent rule flow graph. This stage explicitly targets global reasoning over

non-adjacent rules, enabling ExIde to recover non-linear dependency structures that are common in real-world business regulations.

5 Experiments

In this section, we present a comprehensive evaluation of the **ExIde** framework. Our evaluation focuses on three questions: (1) Does executable grounding provide a stronger inductive signal for business rule extraction? (2) Are reasoning-oriented models necessary for recovering global rule dependencies? (3) How robust is ExIde under increasing logical complexity?

5.1 Experimental Setup

Models. We conduct a comprehensive evaluation across 13 LLMs, spanning both closed-source and open-source architectures, across 5 model families: GPT, Gemini, Kimi, DeepSeek, and Qwen, which cover various model sizes. For Stage I, we assess structured rule extraction using NER F1 for *Slot Type*, *Reference Value*, and *Action*, and F1 for *Logical Operator* classification. For Stage II, dependency identification is formulated as a multi-class classification problem over *Sequential*, *Conditional*, *Parallel*, and *None* relations, evaluated using F1 score.

5.2 Stage I Results: Executable Grounding as a Structural Scaffold

Table 2 and Table 3 report the performance of five prompting strategies across all models. A consistent trend emerges: **Executable Grounding (P5)** achieves the strongest performance, with the highest average F1 scores for both NER and logical operator classification.

This result supports our hypothesis that introducing an intermediate executable representation provides an effective inductive signal for logic-intensive extraction. By forcing early resolution of conditional branches and control flow, pseudo-code grounding helps models produce more precise and structurally consistent outputs.

In contrast, **Logic-Aware Definition Injection (P4)** performs worst on average. Despite explicitly providing logical definitions, P4 degrades performance under dense inputs, suggesting that verbose declarative constraints increase prompt-level reasoning overhead and interfere with the model's ability to maintain consistent rule instantiation under dense inputs. These results indicate that procedural representations are more effective than textual definitions for guiding logical reasoning.

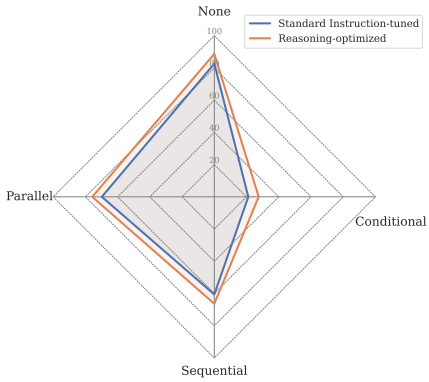

Figure 3: Performance comparison across dependency types using a radar plot. Reasoning-optimized models consistently outperform instruction-tuned baselines.

5.3 Stage II Results: Reasoning Models Bridge the Logic Gap

Table 4 summarizes dependency identification performance. Overall, models equipped with explicit reasoning mechanisms consistently outperform standard instruction-tuned counterparts, highlighting a clear Reasoning Gap. Dependency identification requires tracing causal relations across non-adjacent rules. While instruction-tuned models often rely on shallow semantic similarity, reasoning-oriented models better simulate execution paths, enabling accurate recovery of long-range and non-linear dependencies. This trend is further illustrated in Figure 3, which provides a dependency-type–wise comparison. The performance gains of reasoning models are most pronounced on **non-linear dependencies**, particularly *Conditional* and *Parallel* relations, whereas improvements on linear *Sequential* relations are comparatively modest. This pattern suggests that explicit reasoning mechanisms are especially effective in handling branching and concurrent logic structures, rather than purely linear execution flows. This advantage is most pronounced in smaller models: **Qwen3-30B-Thinking** (0.485) achieves a substantial **+6.0% improvement** over its Instruct counterpart (0.425), suggesting that reasoning capabilities can effectively compensate for model size constraints

5.4 Analysis on Complexity and Robustness

To assess the scalability of ExIde, we analyze performance under increasing complexity dimensions: **Rule Density** (number of rules per document) for business rule extraction, and (2) **Reasoning Com-**

Model	P1	P2	P3	P4	P5	Avg
Closed source						
Gemini-2.5-flash	0.888	0.845	0.873	0.891	0.875	0.874
Gemini-2.5-pro	0.916	0.913	0.907	0.892	0.899	**0.905**
GPT-5-mini	0.794	0.811	0.789	0.824	0.806	0.809
GPT-5	0.876	0.88	0.877	0.802	0.873	0.862
Open source						
Kimi-k2-Instruct	0.908	0.905	0.899	0.883	0.915	0.902
Kimi-k2-Thinking	0.892	0.905	0.889	0.884	0.902	0.894
DeepSeek-3.1	0.883	0.896	0.879	0.884	0.896	0.888
DeepSeek-3.2-exp	0.902	0.909	0.898	0.895	0.905	0.902
DeepSeek-R1	0.885	0.867	0.876	0.875	0.875	0.876
Qwen3-30B-Instruct	0.885	0.886	0.879	0.870	0.880	0.880
Qwen3-30B-Thinking	0.810	0.839	0.805	0.859	0.873	0.837
Qwen3-235B-Instruct	0.898	0.888	0.895	0.888	0.892	0.892
Qwen3-235B-Thinking	0.885	0.877	0.878	0.880	0.881	0.880
Avg	0.879	0.879	0.873	0.871	**0.882**	

Table 2: Rule extraction performance (NER F1) across five prompting strategies. **P1–P5** correspond to five prompting strategies, respectively. **Avg** denotes the average F1 score over all models or over all promptings. **Darker** shades indicate higher performance.

Model	P1	P2	P3	P4	P5	Avg
Closed source						
Gemini-2.5-flash	0.892	0.818	0.864	0.803	0.896	0.855
Gemini-2.5-pro	0.914	0.876	0.838	0.811	0.931	**0.874**
GPT-5-mini	0.793	0.824	0.831	0.796	0.818	0.812
GPT-5	0.838	0.879	0.880	0.755	0.843	0.839
Open source						
Kimi-k2-Instruct	0.843	0.876	0.851	0.767	0.838	0.835
Kimi-k2-Thinking	0.824	0.882	0.874	0.777	0.837	0.839
DeepSeek-3.1	0.822	0.882	0.871	0.776	0.848	0.840
DeepSeek-3.2-exp	0.865	0.896	0.895	0.800	0.848	0.861
DeepSeek-R1	0.855	0.865	0.816	0.778	0.845	0.832
Qwen3-30B-Instruct	0.820	0.834	0.813	0.735	0.811	0.803
Qwen3-30B-Thinking	0.740	0.720	0.644	0.724	0.826	0.731
Qwen3-235B-Instruct	0.837	0.818	0.838	0.771	0.838	0.820
Qwen3-235B-Thinking	0.882	0.834	0.841	0.755	0.867	0.836
Avg	0.840	0.846	0.835	0.773	**0.850**	

Table 3: Logical operator classification performance (F1) under different prompting strategies. **P1–P5** follow the same prompting design as in Table 2. **Avg** denotes the average F1 score over all models or over all promptings.

plexity (number of rule pairs) for dependency relationship identification.

We group test documents by the number of rules they contain, a proxy for information density, and analyze the performance trends of different prompting strategies in the *left* panel of Figure 4. As rule density increases, all methods exhibit performance degradation; however, **P5** shows a relatively flat performance curve, maintaining stable performance even in "high" dense documents containing more than 10 rules. This robustness indicates that executable grounding serves as an effective structural anchor, mitigating the "lost-in-the-middle" effect (He et al., 2024) in long regulatory texts. In contrast, **P4** exhibits a heavy-tailed variance distribution in the *Med-Low* bucket. Case analysis (Appendix A.7) reveals that when business texts enumerate multiple options leading to semantically similar or identical downstream actions, models prompted by P4 frequently over-merge distinct equality conditions into a single *equal* rule. This behavior violates intended atomic semantics of equality and collapses multiple rule instantiations into an abstract representation. Consequently, some models preserve fine-grained rule coverage and achieve near-perfect scores, while others suffer sharp recall degradation, resulting in observed heavy-tailed variance under string-based evaluation metrics.

Although **P5** achieves the best overall accuracy, we observe a bucket-specific reversal for a subset of models in high-density documents. This phenomenon arises from two competing effects (see Appendix A.7). On the one hand, the pseudo-code scaffold in P5 introduces additional prompt overhead and encourages the reification of intermediate meta-level variables, which can generate spurious

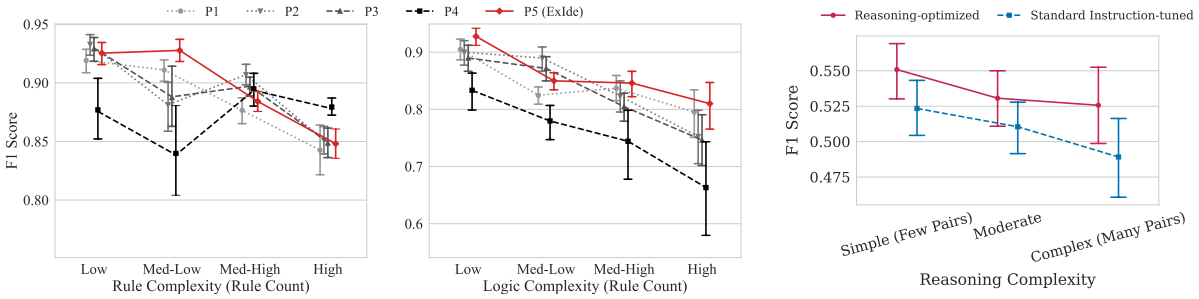

Figure 4: Robustness and Complexity Analysis. The *left* figures show the performance under varying rule and logic complexities, while the *right* figure illustrates the impact of reasoning complexity.

Model	F1
Closed source	
Gemini-2.5-Flash	0.629
Gemini-2.5-Pro	**0.764**
GPT-5-mini	0.550
GPT-5	0.589
Open source	
Kimi-k2-Instruct	0.482
Kimi-k2-Thinking	0.500
DeepSeek-3.1	0.575
DeepSeek-3.2-exp	**0.601**
DeepSeek-R1	0.576
Qwen3-30B-Instruct	0.425
Qwen3-30B-Thinking	0.485
Qwen3-235B-Instruct	0.554
Qwen3-245B-Thinking	0.586

Table 4: Dependency relationship identification performance (F1) of 13 LLMs. This task requires global reasoning over rule-to-rule dependencies (Sequential, Conditional, Parallel, None).

rule tuples and reduce precision in densely coupled workflows. On the other hand, P4's explicit decomposition of *contain* versus *equal* relations can be advantageous in texts dominated by large enumerations or threshold-based rules, as it encourages models to remain grounded in explicit surface entities rather than synthesizing abstract state variables. However, this advantage comes at the cost of increased variance. By introducing an additional decomposition stage, P4 becomes highly sensitive to action-scope interpretation and surface-form drift in action phrases, leading to inconsistent behavior across models and instances. Taken together, these effects explain both the higher variance of P4 and its occasional performance advantage in the High-density bucket.

For dependency reasoning (The right panel of Figure 4), performance divergence becomes more pronounced as the number of rule pairs grows. Standard instruction-tuned models collapse under high reasoning complexity, whereas reasoning-oriented models exhibit significantly flatter degradation curves. These results demonstrate that global rule flow modeling is not merely a pattern-matching task, but requires explicit state tracking and structured reasoning.

6 Conclusion

In this work, we revisited business process modeling from a rule-centric perspective and identified a fundamental limitation of existing approaches: action-centric formulations fail to capture the complex logical dependencies that govern real-world business regulations. To bridge this *Logic Gap*, we introduced **BREX**, a cross-domain benchmark with expert-annotated condition–action rules and explicit rule-to-rule dependencies, enabling systematic evaluation of logic-aware extraction. Building on this benchmark, we proposed **ExIde**, a structure-aware framework that decomposes business rule flow modeling into atomic rule extraction and global dependency reasoning. Through extensive experiments across 13 large language models, we demonstrated that executable grounding provides a strong inductive bias for logic-intensive rule extraction, while reasoning-oriented models consistently outperform instruction-tuned counterparts in recovering long-range and non-linear dependencies. Our findings suggest that effective modeling of complex business logic requires moving beyond surface-level semantic extraction toward execution-oriented representations that align with structured reasoning. We hope BREX will serve as a foundation for future research on logic-aware language models and advance the development of robust, automated process understanding systems.

Limitations

While our work makes progress toward logic-aware business rule flow modeling, several limitations remain.

Synthetic Data Usage. Although the majority of the BREX dataset is derived from real-world business documents, a limited amount of synthetic data is used to augment underrepresented logical structures such as deeply nested conditions and parallel constraints. Despite rigorous expert filtering and revision, synthetic texts may not fully capture the stylistic and contextual nuances of naturally occurring regulatory language. Future work could further reduce reliance on synthetic augmentation by expanding data collection in high-complexity domains.

Prompt-Based Dependency Modeling. ExIde relies on prompt-based reasoning to identify rule-to-rule dependencies rather than an end-to-end trained model. While this design choice enables flexible evaluation across diverse LLMs and avoids task-specific training, it may limit scalability when applied to extremely large rule sets due to the quadratic growth of rule pairs. Incorporating structural pruning or hybrid neural-symbolic approaches could improve efficiency in large-scale deployments.

Scope of Logical Formalization. Our annotation schema focuses on three primary dependency types (Sequential, Conditional, and Parallel) and a predefined set of logical operators. Although these categories cover the majority of business logic observed in our corpus, more expressive constructs—such as temporal constraints, exception handling, or probabilistic rules—are not explicitly modeled. Extending the schema to support richer logical formalisms remains an important direction for future research.

Despite these limitations, we believe our work represents an important step toward bridging natural language business rules and executable logical representations, and we hope it will stimulate further research on logic-aware reasoning with LLMs.

Ethical Statement

This work introduces BREX, a benchmark for business rule flow modeling, and proposes ExIde, a structure-aware reasoning framework for extracting structured business rules and their logical dependencies from unstructured regulatory and administrative texts. We discuss the ethical considerations associated with data collection, annotation, and potential downstream use.

Data Sources and Privacy. The BREX dataset is constructed primarily from publicly available or institutionally released business and regulatory documents, such as administrative guidelines, service manuals, and compliance descriptions. These documents do not contain personal identifiable information or sensitive user data. Any synthetic texts used for data augmentation are generated solely to enrich underrepresented logical structures and are carefully reviewed and revised by domain experts to ensure consistency with real-world regulatory language, without introducing fictitious entities or personal information.

Annotation Process and Human Involvement. All annotations in BREX are performed by trained domain experts following a clearly defined and documented annotation schema. Annotators are instructed to focus exclusively on extracting explicit business rules and logical dependencies present in the text, rather than inferring implicit intent or subjective interpretations. Inter-annotator agreement is evaluated to ensure consistency and reduce individual bias. No crowd-sourced or vulnerable populations are involved in the annotation process.

Potential Misuse and Limitations. While the proposed framework aims to support process automation and regulatory analysis, we acknowledge that automated extraction of business rules may be misused if deployed without appropriate human oversight, particularly in high-stakes domains such as finance, healthcare, or public administration. Incorrectly extracted or interpreted rules could lead to flawed downstream decisions. Therefore, we emphasize that BREX and ExIde are intended as research tools for benchmarking and model analysis, rather than as fully autonomous decision-making systems.

Model Behavior and Bias. Our study evaluates large language models using prompt-based methods without task-specific fine-tuning. As such, model outputs may reflect limitations or biases inherent in the underlying pretrained models. We do not claim that the extracted rule flows are universally correct or legally binding. Instead, our evaluation focuses on structural accuracy relative to expert-annotated ground truth, highlighting both

strengths and failure modes to encourage transparent and responsible model development.

Broader Impact. We believe this work contributes positively to research on logic-aware language understanding and structured reasoning, by providing a realistic benchmark and systematic analysis of model behavior under increasing logical complexity. We hope that BREX will support future research on interpretable, verifiable, and human-in-the-loop systems, where automated rule extraction serves as an assistive tool rather than a replacement for expert judgment.

References

Lars Ackermann, Julian Neuberger, and Stefan Jablonski. 2021. Data-driven annotation of textual process descriptions based on formal meaning representations. In *International Conference on Advanced Information Systems Engineering*, pages 75–90. Springer.

Ron Artstein. 2017. Inter-annotator agreement. *Handbook of linguistic annotation*, pages 297–313.

Patrizio Bellan, Mauro Dragoni, and Chiara Ghidini. 2022a. Extracting business process entities and relations from text using pre-trained language models and in-context learning. In *International Conference on Enterprise Design, Operations, and Computing*, pages 182–199. Springer.

Patrizio Bellan, Chiara Ghidini, Mauro Dragoni, Simone Paolo Ponzetto, and Han van der Aa. 2022b. Process extraction from natural language text: the PET dataset and annotation guidelines. In *Proceedings of the Sixth Workshop on Natural Language for Artificial Intelligence (NL4AI 2022) co-located with 21th International Conference of the Italian Association for Artificial Intelligence (AI*IA 2022), Udine, November 30th, 2022*, volume 3287 of *CEUR Workshop Proceedings*, pages 177–191.

Diogo S Candido, Joao Victor Berti Lima, Hilário Oliveira, and Mateus B Costa. 2024. An annotated dataset for automatic extraction of entities and restrictions from business process models. In *Encontro Nacional de Inteligência Artificial e Computacional (ENIAC)*, pages 978–989. SBC.

Gheorghe Comanici, Eric Bieber, Mike Schaekermann, Ice Pasupat, Noveen Sachdeva, Inderjit Dhillon, Marcel Blistein, Ori Ram, Dan Zhang, Evan Rosen, and 1 others. 2025. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *arXiv preprint arXiv:2507.06261*.

Weihong Du, Wenrui Liao, Hongru Liang, and Wenqiang Lei. 2024. PAGED: A benchmark for procedural graphs extraction from documents. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2024, Bangkok, Thailand, August 11-16, 2024*, pages 10829–10846. Association for Computational Linguistics.

Elena Viorica Epure, Patricia Martín-Rodilla, Charlotte Hug, Rebecca Deneckère, and Camille Salinesi. 2015. Automatic process model discovery from textual methodologies. In *2015 IEEE 9th international conference on research challenges in information science (RCIS)*, pages 19–30. IEEE.

Fabian Friedrich, Jan Mendling, and Frank Puhlmann. 2011. Process model generation from natural language text. In *Advanced Information Systems Engineering: 23rd International Conference, CAiSE 2011, London, UK, June 20-24, 2011. Proceedings 23*, pages 482–496. Springer.

Wenyan Guo, Qingtian Zeng, Hua Duan, Guiyuan Yuan, Weijian Ni, and Cong Liu. 2018. Automatic extraction of emergency response process models from chinese plans. *IEEE Access*, 6:74104–74119.

Junqing He, Kunhao Pan, Xiaoqun Dong, Zhuoyang Song, LiuYiBo LiuYiBo, Qianguosun Qianguosun, Yuxin Liang, Hao Wang, Enming Zhang, and Jiaxing Zhang. 2024. Never lost in the middle: Mastering long-context question answering with position-agnostic decompositional training. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 13628–13642.

Humam Kourani, Alessandro Berti, Daniel Schuster, and Wil MP van der Aalst. 2024. Process modeling with large language models. In *International Conference on Business Process Modeling, Development and Support*, pages 229–244. Springer.

Hongru Liang, Jia Liu, Weihong Du, Dingnan Jin, Wenqiang Lei, Zujie Wen, and Jiancheng Lv. 2023. Knowing-how & knowing-that: A new task for machine comprehension of user manuals. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 10550–10564, Toronto, Canada. Association for Computational Linguistics.

Hugo A López, Rasmus Strømsted, Jean-Marie Niyodusenga, and Morten Marquard. 2021. Declarative process discovery: Linking process and textual views. In *International Conference on Advanced Information Systems Engineering*, pages 109–117. Springer.

Hirokuni Maeta, Tetsuro Sasada, and Shinsuke Mori. 2015. A framework for procedural text understanding. In *Proceedings of the 14th International Conference on Parsing Technologies*, pages 50–60.

Carolyn R Miller. 1979. A humanistic rationale for technical writing. *College English*, 40(6):610–617.

Kuntal Kumar Pal, Kazuaki Kashihara, Pratyay Banerjee, Swaroop Mishra, Ruoyu Wang, and Chitta Baral. 2021. Constructing flow graphs from procedural cybersecurity texts. *arXiv preprint arXiv:2105.14357*.

Vira Pyrih, Adrian Rebmann, and Han van der Aa. 2025. Llms that understand processes: Instruction-tuning for semantics-aware process mining. In *7th International Conference on Process Mining, ICPM 2025, Montevideo, Uruguay, October 20-24, 2025*, pages 1–8. IEEE.

Chen Qian, Lijie Wen, Akhil Kumar, Leilei Lin, Li Lin, Zan Zong, Shu'ang Li, and Jianmin Wang. 2020. An approach for process model extraction by multi-grained text classification. In *Advanced Information Systems Engineering: 32nd International Conference, CAiSE 2020, Grenoble, France, June 8–12, 2020, Proceedings 32*, pages 268–282. Springer.

Luis Quishpi, Josep Carmona, and Lluís Padró. 2020. Extracting annotations from textual descriptions of processes. In *Business Process Management: 18th International Conference, BPM 2020, Seville, Spain, September 13–18, 2020, Proceedings 18*, pages 184–201. Springer.

Andrei Cosmin Redis, Mohammadreza Fani Sani, Bahram Zarrin, and Andrea Burattin. 2024. Processt-bench: An LLM plan generation dataset for process mining. *CoRR*, abs/2409.09191.

Haopeng Ren, Yushi Zeng, Yi Cai, Bihan Zhou, and Zetao Lian. 2023. Constructing procedural graphs with multiple dependency relations: a new dataset and baseline. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 8474–8486.

Patrick E Shrout and Joseph L Fleiss. 1979. Intraclass correlations: uses in assessing rater reliability. *Psychological bulletin*, 86(2):420.

S Sivasankari, Dinah Punnoose, and D Krishnamoorthy. 2020. A comparative study on the performance of rule engines in automated ontology learning: a case study with erythemato-squamous disease (esd). *International Journal of Intelligent Unmanned Systems*, 8(4):267–280.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, and 1 others. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

Petia Wohed, Wil MP van der Aalst, Marlon Dumas, Arthur HM ter Hofstede, and Nick Russell. 2006. On the suitability of bpmn for business process modelling. In *International conference on business process management*, pages 161–176. Springer.

Jianhua Zhang, Muhammad Usman Shehzad, Sher Ali, and Ziao Cao. 2025. Unlocking digital innovation: a moderated-mediation approach exploring the knowledge creation processes, it-enabled capabilities and absorptive capacity in software smes. *Business Process Management Journal*, 31(1):170–201.

A Appendix

A.1 Definition

Definition 1 A **Business Text** is a natural language description of a business scenario or process. It serves to standardize operations and management requirements within a specific domain.

Definition 2 **Business Rules** are explicit instructions describing the conditions, actions, and restrictions that must be followed. A business text typically contains multiple rules. We represent a rule as a binary tuple: ⟨Condition, Action⟩. Although complex rules may involve multiple conditions, we normalize them into atomic condition-action pairs linked through explicit dependency relations, enabling compositional modeling of complex logic. Beyond individual representation, capturing the **Dependency Relationships** between rules is crucial, as they dictate the logical execution flow (e.g., the execution of Rule B depends on the outcome of Rule A).

Based on these definitions, our annotation schema consists of three primary components:

Definition 3 **Condition** specifies the trigger for a business rule, comprising three key attributes: ⟨Slot Type, Logical Operator, Reference Value⟩.

- **Slot Type**: Represents the category of user input (e.g., *currency type*).
- **Logical Operator**: Defines the relation between the slot type and reference value. Permissible values include: *contains*, *equal to*, *less than*, *greater than*, *less than or equal to*, and *greater than or equal to*.
- **Reference Value**: Denotes specific data associated with the slot type, classified into: (a) **Enumeration type** (discrete values, e.g., *USD, RMB*); (b) **Numeric type** (values for comparison, e.g., *$50,000*).

Definition 4 **Action** specifies the operation to be executed once the condition is met. If a process terminates, the Action value can be *None*.

Definition 5 **Dependency Relationship** describes the logical connection between rules:

- **Sequential Dependency**: Rule A must be executed before Rule B.
- **Conditional Dependency**: Different reference values of Rule A trigger different subsequent rules (branching logic).
- **Parallel Dependency**: Two or more rules must be executed simultaneously.

A.2 Illustrative Examples of Dependency Relationships

In this section, we provide illustrative examples of the three types of dependency relationships: sequential, conditional, and parallel. These examples aim to clarify the definitions discussed earlier, with corresponding business scenarios demonstrating the practical implications of each dependency type:

- **Sequential Dependency:** Our bank supports up to 39 currency types of popular countries or regions around the world, including RMB, USD, JPY, GBP, and HKD. After selecting the appropriate currency, the customer needs to choose the corresponding cash or remittance type based on the type of business to be conducted thereafter. The cash or remittance type includes cash and remittance. Upon completing the cash/remittance type selection, the customer needs to provide the purchase amount...

- **Conditional Dependency:** When choosing a car rental service, the user first needs to select the vehicle type, which can be a sedan, SUV, business vehicle, or RV. If the user chooses a sedan or SUV, they will need to further choose the rental duration, with options like 1 day, 1 week, or 1 month. If the user chooses a business vehicle or RV, they will need to choose the rental purpose, which could be for travel, road trips, or business activities. After selecting the rental duration, the user will also need to choose whether to purchase insurance...

- **Parallel Dependency:** When signing up for an educational training course, the user first needs to choose the course type, such as programming, design, marketing, etc. After selecting the course type, the user needs to simultaneously choose the instruction mode and course duration. The instruction mode includes online courses and in-person courses, and the course duration can be 1 month, 3 months, or 6 months. After selecting the instruction mode and course duration, the user will need to choose the learner's age group, which could be adults or teenagers. then, the user will proceed to the tuition payment...

Figures 5, 6, and 7 visually represent the business processes associated with each type of dependency.

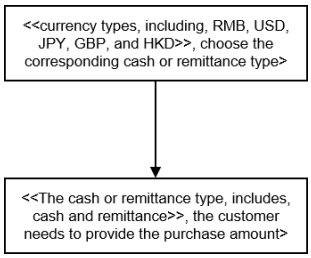

Figure 5: Sequential dependency.

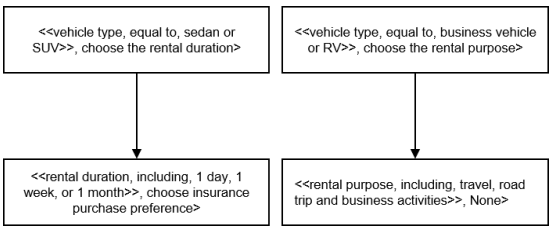

Figure 6: Conditional dependency.

A sequential dependency occurs when the execution of one rule depends on the completion of another rule. In other words, the first rule must be executed before the second. For example, in the case of foreign exchange services, a customer must first select a currency type (such as RMB, USD, JPY, etc.). After selecting the currency, the customer must choose the corresponding cash or remittance type (cash or remittance). Finally, after selecting the cash/remittance type, the customer is required to provide the exchange amount. Each step must occur in a prescribed order, as the actions are dependent on prior selections, as depicted in Fig. 5.

A conditional dependency is present when the next rule to be executed depends on the conditions or choices made in the previous rule. In this scenario, the execution of a rule is conditional on the reference values selected by the user. For instance, in the car rental service example, when a user selects a vehicle type, the subsequent actions depend on the choice made. If the user selects a sedan or SUV, they are prompted to choose the rental duration, with options like 1 day, 1 week, or 1 month. However, if the user selects a business vehicle or RV, they will instead need to choose the rental purpose (e.g., travel, road trip, or business activity). This dependency ensures that the process flow adjusts based on user input, as shown in Fig. 6.

A parallel dependency exists when multiple business rules are executed simultaneously. For example, when registering for an educational training course, a user first selects a course type (such as

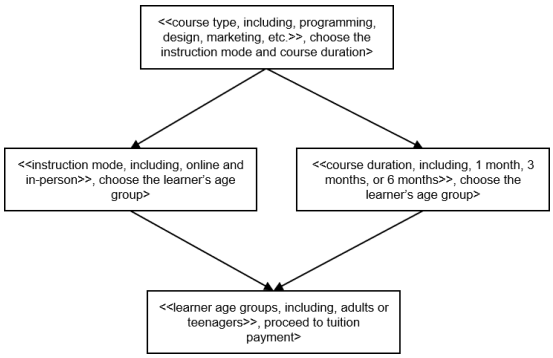

Figure 7: Parallel dependency.

programming, design, marketing, etc.). After this selection, the user must choose both the instruction mode (online or in-person) and the course duration (1 month, 3 months, or 6 months) in parallel. Then, the learner's age group (either adults or teenagers) is selected. Finally, the user proceeds to tuition payment after completing these selections. These choices of instruction mode and course duration occur in parallel, as illustrated in Fig. 7.

A.3 Detailed Dataset Statistics

To provide a granular view of the BREX benchmark, we present detailed statistics in Table 5. The dataset comprises 409 documents with a total of 95,779 tokens and 2,573 sentences, averaging approximately 6.3 sentences and 234 tokens per document. This reflects the concise and information-dense nature of professional regulatory texts.

In terms of **Logical Operators**, the distribution reveals the complexity of real-world constraints. Notably, *Contains* (1,798) and *Greater Than* (>) (900) are the most predominant operators, accounting for the majority of conditions. This indicates that business rules frequently involve set membership checks (e.g., verifying if a product belongs to a specific category) and numerical threshold constraints (e.g., financial limits), rather than simple equality matches.

Regarding **Dependency Relationships**, while *Sequential* dependencies (2,330) form the backbone of most processes, non-linear structures constitute a significant portion of the logic flow. Specifically, *Conditional* (722) and *Parallel* (417) dependencies together account for approximately **33%** of all relations. This high proportion of branching and concurrent logic empirically supports our motivation that modeling business regulations requires capabilities far beyond linear action sequencing.

A.4 Benchmark Comparison

To further clarify the positioning of BREX within the broader landscape of procedural text understanding, we provide a systematic comparison with existing benchmarks in Table 6.

Existing datasets predominantly follow an *Action-Centric* paradigm, treating processes as sequences of events or actions (e.g., recipes, maintenance logs). While some recent works like PAGED (Du et al., 2024) have scaled up the data size significantly, they rely on synthetic data-to-text generation, which often simplifies the linguistic complexity found in real-world regulations. In contrast, BREX is the first benchmark to adopt a *Rule-Centric* modeling approach, explicitly annotating atomic business rules and their complex logical dependencies (Sequential, Conditional, Parallel) across 30+ real-world vertical domains.

A.5 Text Quality Assessment Criteria

To assess the linguistic and practical quality of the business texts in BREX, we conduct a human evaluation along five dimensions: *Readability*, *Accuracy*, *Clarity*, *Simplicity*, and *Usability*. Each dimension is rated independently from 1 to 5 by trained annotators. Below we provide detailed definitions of each dimension, followed by the scoring guidelines used during annotation to ensure consistency and reproducibility. Evaluation results can be seen in Figure 8.

A.6 Details on Inter-Annotator Agreement Calculation

To quantitatively evaluate the reliability of BREX, we addressed the challenge that standard metrics for inter-annotator agreement (IAA) are not directly applicable to complex, graph-structured dependency data. Consequently, we adopted a proxy evaluation strategy by projecting the structured annotations into a flattened Named Entity Recognition (NER) format.

As illustrated in Table 7, we decomposed the hierarchical rule structures into token-level span annotations using the standard BIO (Beginning, Inside, Outside) tagging scheme. We focused on the three atomic components of business rules: **Slot Types**, **Reference Values**, and **Actions**.

The table demonstrates this projection with a representative example. For the phrase *"currency types... including RMB, USD..."*:

- **Span Boundaries**: Annotators must deter-

Docs	Sentences	Rules	Tokens	Logical Operator						Dependency Relationship		
				Contains	Equal	<	>	≤	≥	Sequential	Conditional	Parallel
409	2,573	2,855	95,779	1,798	41	34	900	38	44	2,330	722	417

Table 5: Statistics of the BREX dataset. We present statistical information on the number of documents and various elements in the dataset.

mine the exact boundaries of slots (e.g., "currency" vs. "currency types"). In the example, Annotator 2 marks "types" as `O` (Outside), while Annotators 1 and 3 include it as `I-Slot`.

- **Entity Separation**: Annotators must also distinguish between distinct values. For the enumeration "RMB, USD", Annotator 2 identifies them as separate entities (`B-Ref`, `B-Ref`), representing a fine-grained interpretation, whereas Annotators 1 and 3 might treat the sequence differently.

This linearization transforms the structural agreement problem into a token-level classification problem, enabling the calculation of Fleiss' Kappa (Artstein, 2017). The resulting high agreement score (0.901) confirms that, despite minor boundary variations (as seen in the table), the core semantic definitions of business rules are interpreted consistently across experts.

Evaluation Dimensions.

- *Readability*: The extent to which a business text can be easily read and understood by a general user, considering sentence structure, vocabulary choice, and overall linguistic fluency.
- *Accuracy*: The degree to which the business text correctly and faithfully describes the underlying business service, rules, and constraints, without introducing factual errors, omissions, or misleading information.
- *Clarity*: The extent to which the text presents business rules and execution logic in a clear and logically coherent manner, enabling readers to understand conditional branching, rule dependencies, and overall process flow.
- *Simplicity*: The degree to which the text avoids unnecessary redundancy, excessive verbosity, or irrelevant details, while preserving all essential information required to describe the business process.

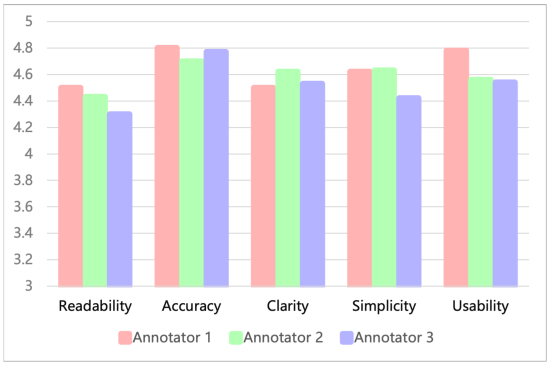

Figure 8: Human evaluation results of text quality across five dimensions among three annotators.

- *Usability*: The effectiveness of the text in guiding users to successfully complete the intended business service or procedure, from an operational and instructional perspective.

All annotators are instructed to apply these criteria consistently across documents and dimensions. Inter-annotator agreement is reported using the Intraclass Correlation Coefficient (ICC) in the main paper, demonstrating high annotation reliability.

A.7 Case Study

Case 1 & Case 2: Over-Merging Enumerated Values under the *equal* Operator. The failures illustrated in Table 8 and Table 9 are caused by an **over-aggressive merging of enumerated values under the *equal* operator**. In Table 8, the business text specifies that after selecting *any* leave type—personal leave, sick leave, or annual leave—the employee must submit the duration of the leave. Although the downstream action is identical, each leave type corresponds to a distinct equality condition and is annotated as a separate rule in the ground truth. In Table 10, the business text explicitly specifies two separate decision branches: after selecting a *standard credit card* or a *themed co-branded card*, the applicant proceeds to the same downstream action. Despite the identical action, the two branches correspond to distinct conditional instantiations and are annotated as separate rules in the ground truth.

Dataset	Samples	Paradigm	Source
Quishpi et al. (2020)	121	Action-Flow	Tutorials, maintenance manuals
Qian et al. (2020)	360	Action-Flow	Cooking recipes and maintenance manuals
Ackermann et al. (2021)	358	Action-Flow	Cooking recipes and maintenance manuals
López et al. (2021)	37	Action-Flow	4 entries from the BPM Academic Initiative
Bellan et al. (2022b)	45	Action-Flow	Academic, Industry, Textbook, Public Sector
Liang et al. (2023)	200	Action-Flow	User manuals, e-commercial helping documents
Ren et al. (2023)	283	Action-Flow	Online wikiHow knowledge base
Du et al. (2024)	3,394	Action-Flow	**Synthetic** (Data-to-Text)
BREX (Ours)	409 (2,855 rules)	Rule-Flow (Condition-Action)	**Real-world** (+Aug) (30+Domains)

Table 6: Comparison of BREX with existing procedural text and process extraction benchmarks. **Paradigm** distinguishes between Action-Flow (extracting event sequences) and Rule-Flow (extracting logical constraints). **Source** indicates whether the text is naturally occurring (Real) or generated (Synthetic).

Text Tokens	...	currency	types	of	...	popular	including	RMB	USD	JPY	...
Annotator 1	...	B-Slot	I-Slot	O	...	O	O	B-Ref	I-Ref	I-Ref	...
Annotator 2	...	B-Slot	O	O	...	O	O	B-Ref	B-Ref	B-Ref	...
Annotator 3	...	B-Slot	I-Slot	O	...	O	O	B-Ref	I-Ref	I-Ref	...

Table 7: An example of transforming rule annotation into NER format for Inter-Annotator Agreement (IAA) calculation.

Under Prompt4, some models incorrectly collapse all enumerated leave types into a single compound equality condition (e.g., *equal(personal, sick, annual)*). This violates the intended semantics of the rule schema, where the *equal* operator is defined over a single atomic value rather than a set. As a result, multiple rule instantiations are replaced by a single abstract rule, leading to a loss of rule granularity. While the generated output appears more concise, it fails to preserve the one-to-one correspondence between decision options and rule instances required by the annotation standard, resulting in reduced recall and lower F1 scores.

Case 3: Abstraction-Induced Precision Degradation in High-Complexity Texts. Table 10 presents a representative High-complexity example involving a dense fixed-deposit process with multiple branching conditions, numeric thresholds, and auxiliary services, resulting in a high rule count (15 rules). In this setting, Prompt 4 consistently grounds its outputs in **explicit, surface-level slots and thresholds directly stated in the text**, such as deposit type, deposit amount constraints, automatic renewal options, and notification methods. By discouraging abstraction beyond textually grounded entities, Prompt 4 produces rule sets that closely align with the annotated schema and maintain high precision.

In contrast, Prompt 5's pseudo-code paradigm introduces a distinct failure mode under high logical density. By enforcing a procedural abstraction, Prompt 5 encourages the model to **reify discourse-level connective phrases**—such as "once the above information is confirmed" or "these two services can be selected simultaneously"—into variables or slot-like entities. This results in the generation of abstract or meta-level rules (e.g., slots like *"above information"* or composite service entities) that do not exist in the ground truth annotation. While such rules may appear structurally coherent within the pseudo-code framework, they constitute spurious extractions under tuple-level evaluation and directly reduce precision.

This phenomenon is amplified in high-density texts, where narrative connectors and summarizing statements are more frequent. As a result, Prompt 5 systematically over-generates abstract slots and meta-rules, leading to a pronounced precision drop, particularly for smaller or less robust models. Prompt 4, by contrast, retains a more conservative extraction bias, favoring concrete, text-aligned rules over procedural abstraction. This conservative bias enables Prompt 4 to outperform Prompt 5 in the High bucket, despite its higher variance in lower-complexity regimes.

Category	Content
Input Text	When employees apply for leave in the company, they first need to clarify the type of leave, which mainly includes personal leave, sick leave and annual leave. After choosing the type of leave, employees need to submit the duration of their leave.
Ground Truth	1. ⟨⟨Type of leave, include, personal leave, sick leave and annual leave⟩, None⟩ 2. ⟨⟨Type of leave, equal, personal leave⟩, employees need to submit the duration of their leave⟩ 3. ⟨⟨Type of leave, equal, sick leave⟩, employees need to submit the duration of their leave⟩ 4. ⟨⟨Type of leave, equal, annual leave⟩, employees need to submit the duration of their leave⟩
Good Output ✅	1. ⟨⟨Type of leave, include, personal leave, sick leave and annual leave⟩, None⟩ 2. ⟨⟨Type of leave, equal, personal leave⟩, employees need to submit the duration of their leave⟩ 3. ⟨⟨Type of leave, equal, sick leave⟩, employees need to submit the duration of their leave⟩ 4. ⟨⟨Type of leave, equal, annual leave⟩, employees need to submit the duration of their leave⟩
Bad Output ✕	1. ⟨⟨Type of leave, include, personal leave, sick leave and annual leave⟩, None⟩ 2. ⟨⟨Type of leave, equal, personal leave, sick leave and annual leave⟩, employees need to submit the duration of their leave⟩

Table 8: Case 1: Over-merging enumerated values under the *equal* operator. Distinct leave types with identical downstream actions are incorrectly collapsed into a single equality condition, violating the atomic semantics of equality and reducing rule granularity.

Category	Content
Input Text	When applying for our bank's credit card, the applicant needs to first select the type of card they wish to apply for. Our bank offers two options: standard credit cards and themed co-branded cards. After choosing the standard credit card, the applicant needs to fill in their personal basic information. ... After choosing the theme co-branded card, the applicant also needs to fill in their personal basic information.
Ground Truth	1. ⟨⟨Type of card, include, standard credit cards and themed co-branded cards⟩, None⟩ 2. ⟨⟨Type of card, equal, standard credit card⟩, the applicant needs to fill in their personal basic information⟩ 3. ⟨⟨Type of card, equal, theme co-branded card⟩, the applicant also needs to fill in their personal basic information⟩
Good Output ✅	1. ⟨⟨Type of card, include, standard credit cards and themed co-branded cards⟩, None⟩ 2. ⟨⟨Type of card, equal, standard credit card⟩, the applicant needs to fill in their personal basic information⟩ 3. ⟨⟨Type of card, equal, theme co-branded card⟩, the applicant also needs to fill in their personal basic information⟩
Bad Output ✗	1. ⟨⟨Type of card, include, standard credit cards and themed co-branded cards⟩, None⟩ 2. ⟨⟨Type of card, equal, standard credit card, theme co-branded card⟩, the applicant needs to fill in their personal basic information⟩

Table 9: Case 2: Over-merging enumerated values under the *equal* operator. Distinct card types with identical downstream actions are incorrectly collapsed into a single equality condition, violating the atomic semantics of equality and reducing rule granularity.

Category	Content
Input Text	When handling a time deposit at a bank, the customer must first choose the type of deposit, which mainly includes two options: lump-sum deposit with lump-sum withdrawal and installment deposit with lump-sum withdrawal. If the customer chooses a lump-sum deposit with lump-sum withdrawal, they must first determine the deposit currency. The bank supports currencies including the Chinese yuan (RMB), U.S. dollars, and euros. Next, the customer selects the deposit term, which can be one year, three years, or five years. If the customer chooses an installment deposit with lump-sum withdrawal, they must first set a fixed monthly deposit amount, which must be no less than 50 yuan. After that, the customer selects the deposit term, which can be either one year or three years. Once the above information is confirmed, the customer needs to provide the specific deposit amount. If a single deposit exceeds 200,000 yuan, the customer must additionally provide proof of the source of funds; if the amount is less than or equal to 200,000 yuan, no such proof is required. Finally, the customer may choose whether to activate the automatic renewal service and the method for receiving interest rate change notifications. The automatic renewal service can be set to "Yes" or "No," while notification methods include "SMS notification" and "email notification." These two services can be selected simultaneously.
Ground Truth	1. ⟨⟨ Type of deposit, include, lump-sum deposit with lump-sum withdrawal and installment deposit with lump-sum withdrawal⟩, None⟩ 2. ⟨⟨ Monthly deposit amount, >, 200,000⟩, the customer must additionally provide proof of the source of funds⟩ 3. ⟨⟨ Automatic renewal service, include, "Yes" or "No,"⟩, None⟩ 4. ⟨⟨ Notification methods, include, "SMS notification" and "email notification."⟩, None⟩ ...
Prompt 4 - Good Output ✅	1. ⟨⟨ Type of deposit, include, lump-sum deposit with lump-sum withdrawal and installment deposit with lump-sum withdrawal⟩, None⟩ 2. ⟨⟨ Monthly deposit amount, >, 200,000⟩, the customer must additionally provide proof of the source of funds⟩ 3. ⟨⟨ Automatic renewal service, include, "Yes" or "No,"⟩, None⟩ 4. ⟨⟨ Notification methods, include, "SMS notification" and "email notification."⟩, None⟩ ...
Prompt 5 - Bad Output ✕	1. ⟨⟨ Above information, include, type of deposit, the deposit currency, deposit term and monthly deposit amount⟩, customer needs to provide the specific deposit amount⟩ 2. ⟨⟨ The automatic renewal service and the method for receiving interest rate change notifications, include, automatic renewal service, notification methods⟩, These two services can be selected simultaneously.⟩ ...

Table 10: Case 3: Abstraction-induced precision degradation in high-complexity texts. Prompt 5 reifies discourse-level connectors into abstract slots and meta-rules, leading to spurious extractions, while Prompt 4 remains grounded in text-aligned fields and achieves higher precision.

##Role
You are a master of logic and deduction, able to deduce the interdependence and logical relationship between different things. You are good at factual summaries, situational analyses, and are both realistic and creative.

##Background
I 'm going to give you an example that contains many fields, represented by the following array: [Intent, Slot Type, Dependency, Structure, Business Rule Pair, Text]
I will give you a literal explanation of these fields, but the **deeper meaning** is something you need to understand on your own! I 'll also give you hints about the logical relationships between these fields, but the **deeper logical relationships** need to be reasoned out by yourself.

Explanation
Intent: the purpose achieved in the process text, i.e. the intent, which expresses different business scenarios.
Slot: the classification used to capture and store user input in the dialogue system, each slot type corresponds to a specific information category.
Text: the relevant business description corresponding to the intent.
Business Rule Pair : all the rules contained inside the text.
Dependency: indicates the dependency relationship between rules.
Structure: sequential structure, parallel structure, selective structure.

##Definitions
{***Definition 1,2,3,4,5***}

Process
1.** Generate business process text **, please cut from different areas of the business process, can contain the financial industry, law and other large industries, details can be to the banking business, real estate business, and a variety of service industry business, business processes can not be too general, abstract, ambiguous, it must be detailed, specific, and at the same time must be in line with the logic of real life, China 's actual national conditions and social details. After generating you need to check again to make sure the statements are smooth and in line with the logic of reality.
2.** Find out the business rules based on the business text**, find out the business rules in the business process text, the form of the business rules binary group has been given above, the business rules must be in the natural language and logical level completely in line with the business rules text.
3.**Find the slot types from the business rules**, see **some details** above
4.** Identify dependencies and structures**

fewshots

Figure 9: The prompt of synthetic text generation.

##Role
You are a professor of logical reasoning, able to deduce the interdependence and logical relationships between different things, you are good at factual summaries, situational analyses, faithful to reality yet creative.

##Definitions
{***Definition 1,2***}
Condition specifies the trigger for a business rule and comprises three key attributes: <Slot Type, Logical Judgement, Reference Value>, where
Slot Type represents a category of user input captured in a dialog system, corresponding to specific information types such as currency type and customer type. **Requirement**: Slot type must be a noun that exists in the business text. The slot cannot be a verb.
Logical Judgement defines the logical relation between the slot type and the reference value, with permissible values including contains, equal to, less than, greater than, less than or equal to and greater than or equal to.
Reference Value denotes the data associated with the slot type in the business text. **Requirement**: The reference value must be a phrase that exists in the business text and the reference value cannot be repeated. It can be classified into two types:
Enumeration type, representing discrete values such as currency types (e.g., USD, RMB, EUR).
Numeric type, used for numerical comparisons, where a reference value (e.g., $50,000) is subject to logical judgments such as greater than or less than.
{***Definition 4***} **Requirement**: Actions must be sentences that exist in the business text. Do not generate sentences that are unrelated to the business text.

Few shots
{Example1:
{Input: }
Our bank supports up to 39 currency types of popular countries or regions around the world, including RMB, US dollar, Japanese yen, British pound, and Hong Kong dollar. After selecting the appropriate currency, the customer needs to choose the corresponding cash or remittance type based on the type of business to be conducted thereafter. The cash or remittance type includes cash and remittance. For foreign currency cash withdrawals, please select "cash"; for foreign exchange remittances, please select "remittance". Upon completing the cash/remittance type selection, the customer needs to provide the purchase amount...
{**Explain**:}
{**Output**:}
}

Execution
You need to analyse, reason and execute strictly according to the definition. Next I will provide you with a piece of business text, you need to extract the business rule binary contained in the business text I provided and send it to me in the format of the corresponding output in the example.

Figure 10: The basic template of five prompts.

Explain:
1. Our bank supports up to 39 currency types of popular countries or regions around the world, including RMB, USD, JPY, GBP, and HKD. After selecting the appropriate currency, the customer needs to choose the corresponding cash or remittance type based on the type of business to be conducted thereafter.
2. The cash or remittance type includes cash and remittance. For foreign currency cash withdrawals, please select "cash"; for foreign exchange remittances, please select "remittance". Upon completing the cash/remittance type selection
...

Output:
1. << currency types, including, RMB, USD, JPY, GBP, and HKD. >, choose the corresponding cash or remittance type >
2. << The cash or remittance type, includes, cash and remittance. >, the customer needs to provide the purchase amount>
...

Figure 11: The details for Prompt 1 (Implicit Semantic Alignment).

Output:
1. **Explain**: Our bank supports up to 39 currency types of popular countries or regions around the world, including RMB, USD, JPY, GBP, and HKD. After selecting the appropriate currency, the customer needs to choose the corresponding cash or remittance type based on the type of business to be conducted thereafter. **The sentence corresponds to the business rule pair:**
<< currency types, including, RMB, USD, JPY, GBP, and HKD. >, choose the corresponding cash or remittance type >
2. **Explain:**
...

Figure 12: The details for Prompt 2 (Explicit Traceability).

The specific example for ‘contain‘ and ‘equal to‘ is as follows:
Input: we mainly support four business types, a, b, c, d. If you choose a, you need x; if you choose b, c, d, you need y.
Output: <<Business Type, Contains, a, b, c, d>, None>
<<Business type, equals, a >, needs x>
<<Business type, equals, b >, requires y>>
<<Business type, equals, c >, requires y>>
<<Business type, equals, d >, requires y>>
Input: when business type is a, we need x
Output: <<business type, equals, a >, needs x>

Figure 13: The details for Prompt 4 (Logic-Aware Definition Injection).

Input:
First, the user selects the currency type from the supported options, which include RMB, USD, JPY, GBP, and HKD. Following this, the user must choose the cash or remittance type.
Next, the user enters the purchase amount. If the purchase amount is less than or equal to $50,000, the request can be handled directly using a valid ID. However, if the purchase amount exceeds $50,000, the user is required to handle the purchase of foreign exchange with their own valid identity document and provide relevant supporting documents.
Finally, the system determines the customer type. If the customer is a ‘Domestic Individual’, they must select the purpose of the foreign exchange purchase from ‘Outbound Travel’, ‘Study Abroad’, or ‘Other Current Account Uses’. If the customer is an ‘Outbound Individual’, they need to select the source of the foreign exchange purchase, including ‘Currency redemption’, ‘Domestic legal income’, or ‘Other sources’.

The following is the **pseudo-code** that corresponds to this business text:
First, the two functions included in the pseudo-code and their specific roles are described:
select_from: this function is used to display the reference values in the business rule binary from which the user can select a specific option.
execute_action: this function is responsible for executing the action in the business rule binary.

Next, show the **pseudo-code** section:

```
Currency type= select_from([‘RMB’, ‘USD’, ‘JPY’, ‘GBP’, ‘HKD’])
execute_action(‘Select the corresponding currency type’)
The cash or remittance type = select_from([‘cash’, ‘remittance’])
Purchase Amount = execute_action(‘Enter Purchase Amount’)
if Purchase Amount <= $50,000:
  execute_action(‘Directly handle with your valid ID’)
elif Purchase amount >$50,000:
  execute_action(‘Handle the purchase of foreign exchange with your own valid identity document and relevant supporting documents’)
// Select customer type
if Customer Type == ‘Domestic Individual’:
  execute_action(‘Select the purpose of foreign exchange purchase’)
  Use of foreign exchange purchase = select_from([‘Outbound Travel’, ‘Study Abroad’, ‘Other Current Account Uses’])
  execute_action(None)
elif Customer Type == ‘Outbound Individual’.
  execute_action(‘Select source of foreign exchange purchase’)
  Source = select_from([‘Currency redemption’, ‘Domestic legal income’, ‘Other sources’])
  execute_action(None)
```

Figure 14: The details for Prompt 5 (Executable Grounding).

##Role
You are a business process analysis expert, skilled at judging dependency relationships based on business texts and business rules. Your task is: based on the business text and the given pairs of business rules, determine the dependency relationship between each pair (sequential relationship, conditional relationship, parallel relationship, or no relationship).

##Definitions
{***Definition 1,2,3,4***}

##Types of Dependency Relationships
There are four types of dependency relationships:
1. **Sequential Dependency**: Some business rules must be executed in a specific order, and there are no other steps between the two rules.
   -**Key Point**-: Only direct sequential dependencies should be retained—there must be no intermediate steps. For example, Rule A must be executed before Rule B, and there is a clear direct dependency between them.
2. **Conditional Dependency**: Different reference values of a business rule determine which subsequent rule will be executed.
   -**Key Point**-: Different reference values for the same slot type lead to different subsequent rules being chosen. For example, if Rule A's slot type has two different reference values, the system will execute either Rule B or Rule C depending on the value. Rule B and Rule C describe different slot types, not different values of the same slot type. The conditional dependency is between rules, meaning that based on Rule A' s reference value, only one of several rules will be selected and executed—not multiple rules simultaneously.
3. **Parallel Dependency**: Some business rules can be executed at the same time.
   -**Key Point**-: Parallel relationships are often indicated in the business text using words like "simultaneously," "and," or "as well as," e.g., "The customer needs to choose both the vehicle type and the rental period plan."
4. **No Relationship**: Some business rules have no direct dependency.
   -**Key Point**-: Any rule pair that does not belong to sequential, choice, or parallel relationships should be classified as "no relationship."

Input Format:
-**Business Text_Input**-: The textual content describing the business process.
-**Business Rule Pairs_Input**-: A list of rule pairs for which dependency relationships need to be judged, in the format <Rule 1> -> <Rule 2>.

Output Format:
-**Dependency Relationship Judgment Results**-: List the dependency type for each rule pair. The format is <Business Rule 1> -> <Business Rule 2>: Dependency Type.

Key Emphases:
1. Sequential: Only direct sequential dependencies should be kept; no intermediate steps allowed.
2. Conditional: Different reference values of Rule A lead the system to select either Rule B or Rule C; Rule B and Rule C describe different slot types.
3. Parallel: Rules can be executed at the same time, often joined by "and", "simultaneously", or similar terms.
4. No Relationship: Any rule pair not fitting the above three types is classified as "no relationship".

##fewshots { }

input
business text input { }
business rule pairs { }

Figure 15: The details for the prompt of identifying dependency relationships.