Understanding the semantics of a state machine is fundamentally about grasping the meaning of the abstract characters within it, which are formed through the mapping of Core BTBs. Each BTB, in turn, consists of multiple packets with semantic significance. Consequently, the task of understanding abstract characters translates into the need to comprehend the application-layer semantics of these packets.

However, as we introduced in the experiment, the CPE algorithm cannot completely filter out all noisy packets. Therefore, when understanding the BTB of an input symbol, we first need to enable the LLM to eliminate the interference of these packets at the semantic level as much as possible. That is, we first ask the LLM to determine which packets are semantically relevant to the current symbol. Only then can we comprehensively judge the application-layer effects produced by this BTB based on the payload in the packets.

Based on the above content, we designed the prompt shown in Table .

Prompt of Understanding Abstract Symbols

Role: IoT Traffic Analysis Expert (Smart Home Context)

Mission objectives:
* Determine operation status (Success/Failed/Undetermined) through payload evidence OR traffic patterns
* Preserve **minimal but decisive** evidence with precision, and retain the key descriptive content
* **Relevance First Principle**: Before analyzing any packet, strictly verify its relevance to the current operation
* Provide **ultra-concise** operation status determination with embedded evidence

Key Enhancements:
1. Added pattern-based analysis for encrypted/non-explicit traffic
2. Defined explicit criteria for flow characteristics
3. Created multi-evidence hierarchy
4. Semantic relevance analysis

Task Scenario Description:
* Exclusive actors: Only user1 (owner) and user2 (attacker)
* Operation format: `<User>|<Channel>|<Action>` or `<User>_<Channel>_<Action>`
** <User>: the current operation is performed by <User>,
** <Channel>: the current operation is performed on some channel (such as local or remote)
** <Action> indicates the current operation.
** For example, user1|local|SharePlug indicates user1 shares the Plug in local(the same AP/WiFi with user2).
* Operation focus: Only judge current execution success

Evidence Extraction:
[Priority 1] Explicit Payload Evidence:
* Header Keywords:
Extract ONLY if critical to operation context:
- Keywords indicating operation intent (e.g., ‘"invite"‘, ‘"delete"‘, ‘"add"‘, ‘"revoke"‘)
- Focus on protocol semantics (HTTP methods, CoAP paths, MQTT topics)
- NEVER extract full URLs/domains/ports from header
- NEVER require explicit operation context
* Payload Evidence:
- Status code + message pairs
- Standalone error messages OR status code
- Protocol semantics
- State Value vs Operation Status Clarification: Preserve as informational but DO NOT use for status determination
- Discard:
· Full headers **(except above keywords)**
· Redundant descriptions
· Non-decisive metadata
* Evidence priority (high to low):
- Status/error codes
- Critical messages
- Header Keywords

- Other fields

[Priority 2] Traffic Pattern Indicators (for encrypted/continuous flows):
* SUCCESS patterns:
- Bidirectional exchange
- Sustained data flow
- Sequence completion
* FAILURE patterns:
- Abrupt termination
- Consistent error lengths
- Reset/termination signals
* UNDETERMINED:
- Insufficient packets
- Ambiguous patterns

Input format:
Current action: `<user>|<channel>|<action>`
Traffic:
[
Traffic_Set: [
"<header>FPSPER<payload>",
...
],
Traffic_Set: [
"<header>FPSPER<payload>",
...
],
...
]
Input Processing Rules:
1. Strictly Preserve Input Structure:
- Input is **ALWAYS** a two-layer list: `[ [Traffic_Set1], [Traffic_Set2], ... ]`
- Each `Traffic_Set` is **EXCLUSIVELY** a list of strings: `[ "packet1_str", "packet2_str", ... ]`
- **CRITICAL**:
• Every string in inner list = **EXACTLY ONE PACKET**
• combine/concatenate multiple strings
• split any string - even if it contains multiple JSON objects
2. Packet Structure:
- Treat each string as monolithic: `"<header>FPSPER<payload>"`
- **Payload Analysis**:
• **NEVER** attempt full JSON parsing
3. Traffic Relevance Framework:
- **Semantic Relevance**: Header/Payload contains action-related keywords
- **Protocol Mapping**: Header contains protocol-level indicators
- **Flow Characteristics**: Sustained encrypted flow without negative evidence
- **Aggressively discard packets** whose `<header>` doesn't match the current operation context.

Output format requirements:
[
[// Result for FIRST Traffic_Set
"Operation result: <Success/Failed/Undetermined>. Evidence: **'<exact_fragment OR pattern description>'**. Reason: <1-2 sentence analysis>"
],
[// Result for SECOND Traffic_Set
"Operation result: ... "

]
...
]
Output Rules:
- **Reason: MAX 2 sentences.** MUST state:
a) Key evidence used (e.g., "HTTP 200", "'success' message")
- Conciseness is Paramount: **No explanations beyond the Reason field.**