

DSL Design Document

The DSL is engineered to:

- Represent boolean expressions that evaluate to True/False.
- Provide an intuitive rule-writing experience for traders.
- Translate easily into an Abstract Syntax Tree (AST) for program execution.
- Maintain clarity, consistency, and determinism.

This report details the DSL's design philosophy, grammar, token definitions, operator precedence, and examples demonstrating its use.

A. Design Philosophy

The DSL is defined by two top-level statements, **ENTRY:** and **EXIT:**, which are processed sequentially. The grammar is designed to be easily parsed by the system's strategy_parser into a machine-readable **Abstract Syntax Tree (AST)**.

B. DSL Grammar and Syntax

Component	Syntax Format	Description	Example
Top-Level	ACTION: EXPRESSION	Defines the condition for a trade action.	ENTRY: close > SMA(close, 20)
Series	SERIES_NAME	Refers to a raw price/volume column.	close, volume, high
Literals	NUMBER	Fixed numeric values.	100.0, 5000000, 20
Indicators	INDICATOR_NAME (SERIES, PERIOD)	Standard technical analysis functions.	SMA(close, 20), RSI(close, 14)
Comparisons	LHS OP RHS	Compares two components. OP includes: >, <, =, >=, <=.	close > 100
Boolean Logic	EXP AND EXP or EXP OR EXP	Combines two or more comparisons using Boolean operators.	AND, OR

C. Indicator Support

The language supports standard technical analysis indicators using a function-call syntax:

Indicator	Function Signature	Description
Simple Moving Average	SMA(series, period)	The simple average of the series over the specified time period.
Relative Strength Index	RSI(series, period)	A momentum oscillator.

1. Tokenization and Terminal Definitions

The DSL relies on specific regular expressions to tokenize raw input into recognized terminals. This ensures clear boundaries for series names, operators, and indicator functions.

- **Series Definition:** The SERIES terminal accepts standard price/volume names (close, open, high, low, volume) and specifically supports lookback indexing.
 - *Regex:* / (close|open|high|low|volume)(\[0-9]+\)? /
- **Indicator Definition:** The INDICATOR terminal strictly enforces the function-call syntax, requiring a valid name (SMA or RSI), a target series, and a period.
 - *Regex:* / (SMA|RSI)((\[a-z]+\s*,\s*\[0-9]+\))/

2. Handling Lookback Data

The DSL allows referencing historical data points via the SERIES terminal to enable cross-day comparisons (e.g., comparing today's close to yesterday's high).

- **Syntax:** SERIES_NAME[-N]
- **Example:** high[-1] refers to the high price of the previous trading period.
- **Implementation:** The StrategyTransformer extracts the lookback integer (e.g., -1) from the token and stores it in the AST as the lookback attribute for the series node.

3. Operator Precedence and Associativity

The grammar enforces standard Boolean operator precedence to correctly group complex conditions, eliminating ambiguity in the resulting AST:

1. **Grouping:** Parentheses () are evaluated first.
2. **AND Operation:** The AND operator takes precedence over OR.
3. **OR Operation:** The OR operator is evaluated last.

Example: The expression A AND B OR C is interpreted as (A AND B) OR C. The parser ensures the AND operation is grouped before the OR operation is considered. This strict ordering is defined in the `strategyparser.py` grammar rules.

D. Example Translations

This demonstrates how the Natural Language instruction maps directly to the required DSL structure:

NL Input	Corresponding DSL
"Buy when the close price is above the 20-day moving average and volume is above 1 million. Exit when RSI(14) is below 20."	ENTRY: <code>close > SMA(close, 20) AND volume > 1000000</code> EXIT: <code>RSI(close, 14) < 20</code>
"Enter when the high price is below the 50-day moving average OR volume is greater than 5 million."	ENTRY: <code>high < SMA(close, 50) OR volume > 5000000</code> EXIT: <code>0 > 1</code> (Placeholder for no exit rule)