Lark Cheat Sheet

by erezsh via cheatography.com/61630/cs/15903/

Lark Options

`parser="earley"`	Use the Earley parser (default)
`parser="lalr"`	Use the LALR(1) parser
`parser="cyk"`	Use the CYK parser
`lexer="standard"`	Use the standard lexer
`ambiguity='explicit'`	Return all derivations for Earley
`start="foo"`	Use "foo" as starting rule
`transformer=...`	Apply transformer to tree (for LALR)
`propagate_positions`	Fill tree instances with line number information
`keep_all_tokens`	Don't remove unnamed terminals
`postlex`	Provide a wrapper for the lexer
`tree_class`	Provide an alternative for `Tree`

Token Reference

`token.type`	Returns name of terminal
`token.value`	Return matched string
`token.line`	Line of match
`token.column`	Column of match
`token.end_line`	Line where match ends
`token.end_column`	Column where match ends
len(token)	Length of match

Tokens inherit from `str`, so all string operations are valid
(such as `token.upper()`).

Grammar Definitions

`rule: ...`	Define a rule
`TERM: ...`	Define a terminal
`rule.n: ...`	Rule with priority n
`TERM.n: ...`	Terminal with priority n
`// text`	Comment
`%ignore ...`	Ignore terminal in input
`%import ...`	Import terminal from file
`%declare TERM`	Declare a terminal without a pattern (used for postlex)

Rules consist of values, other rules and terminals.
Terminals only consist of values and other terminals.

Grammar Patterns

`foo bar`	Match sequence
`(foo bar)`	Group together (for operations)
`foo | bar`	Match one or the other
`foo?`	Match 0 or 1 instances
`[foo bar]`	Match 0 or 1 instances
`foo*`	Match 0 or more instances
`foo+`	Match 1 or more instances
`foo~3`	Match exactly 3 instances
`foo~3..5`	Match between 3 to 5 instances

Terminal Atoms

`"string"`	String to match
`"string"i`	Case-insensitive string
`/regexp/`	Regular Expression
`/re/imslux`	Regular Expression with flags
`"a".."z"`	Literal range

Tree Shaping

`rule: "foo" BAR`	"foo" will be filtered out
`!rule: "foo" BAR`	"foo" will be kept
`rule: /foo/ BAR`	/foo/ will be kept
`_TERM`	Filter out this terminal
`_rule`	Always inline this rule
`?rule: ...`	Inline if matched 1 child
`foo bar -> alias`	Create alias

Rules are a branch (node) in the resulting tree, and its children are its matches, in the order of matching.
Terminals (tokens) are always values in the tree, never branches.
Inlining rules means removing their branch and replacing it with their children.

Tree Reference

`tree.data`	Get rule name
`tree.children`	Get rule matches
`print(tree.pretty())`	Pretty-print tree
`tree.iter_subtrees()`	Iterate on all nodes
`tree.find_data("foo")`	Find nodes with rule `foo`
`tree.find_pred(...)`	Find nodes by predicate
tree1 == tree2	Compare trees

By **erezsh**
cheatography.com/erezsh/

Not published yet.
Last updated 26th May, 2018.