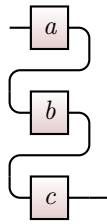
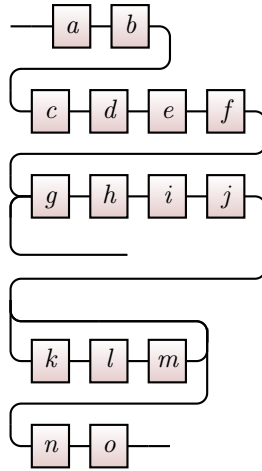


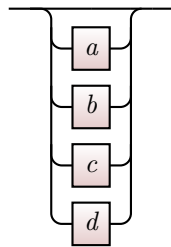
newlines



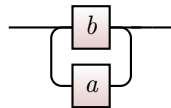
otherb



simple



simple



— (') *graphic_character* (') —

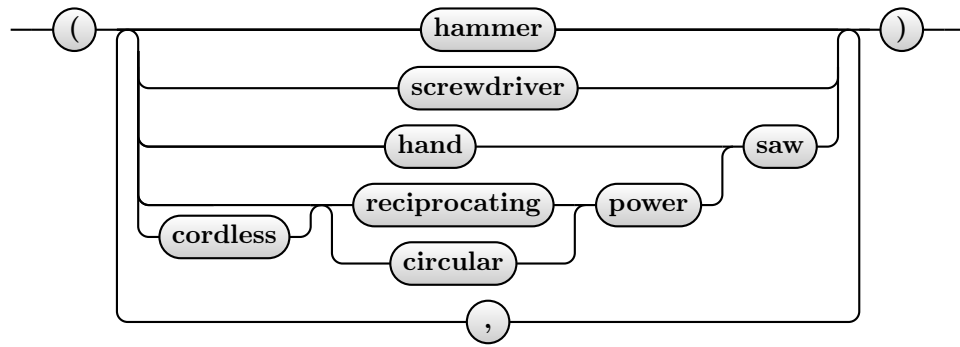
The diagram illustrates the syntax for a `case` statement. It consists of the following components and flow:

- Case Header:** A `case` node (green oval) is followed by an `expression` node (pink rectangle) and an `is` node (green oval).
- Case Alternatives:** A loop structure where a `case_statement_alternative` node (pink rectangle) is followed by a comma node (green circle). This loop is represented by a horizontal line with a vertical drop, a horizontal segment, and a vertical rise.
- Case Body:** An `end` node (green oval) is followed by a `case` node (green oval).
- Case Label:** A `label` node (pink rectangle) is connected to the `case` node in the body by a line that goes down, then right, then up.
- Termination:** The entire statement is terminated by a semicolon node (green circle).

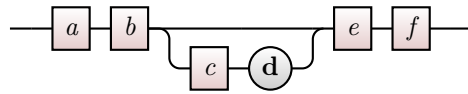
```

graph LR
    when((when)) --- choice1(( ))
    choice1 --- simple_expression[simple_expression]
    choice1 --- discrete_range[discrete_range]
    choice1 --- simple_name[simple_name]
    choice1 --- others((others))
    choice1 --- choice2(( ))
    choice2 --> seq_of_statements[sequence_of_statements]
  
```

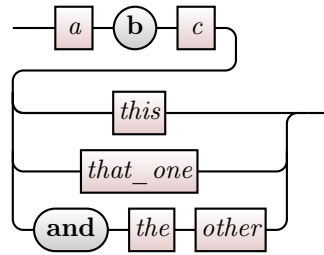
*tool*list



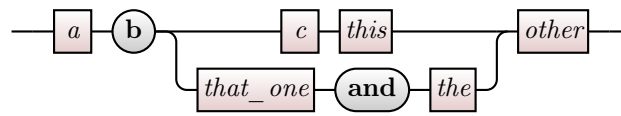
*no*list



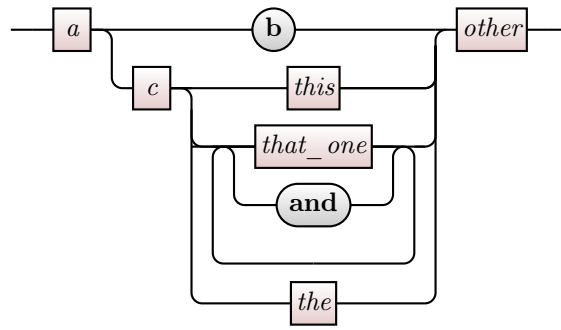
list



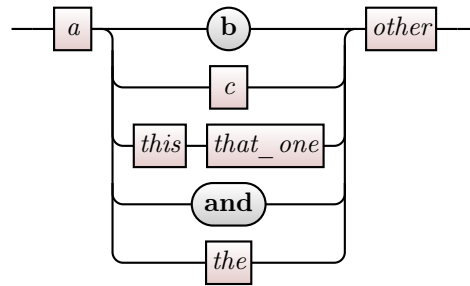
list



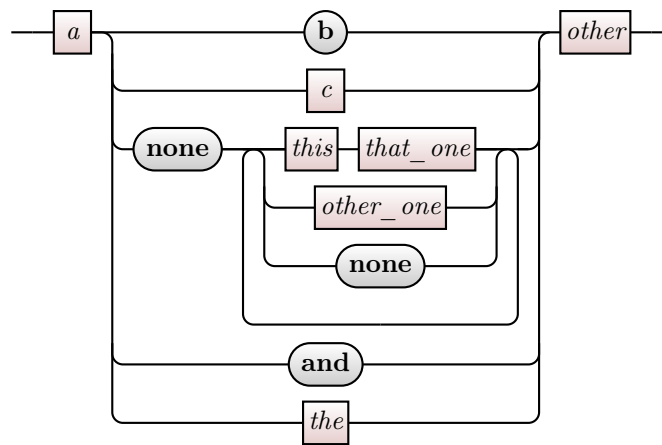
listx



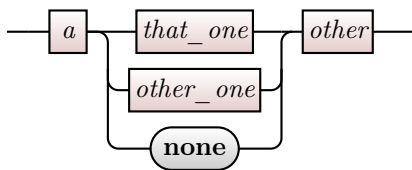
listj



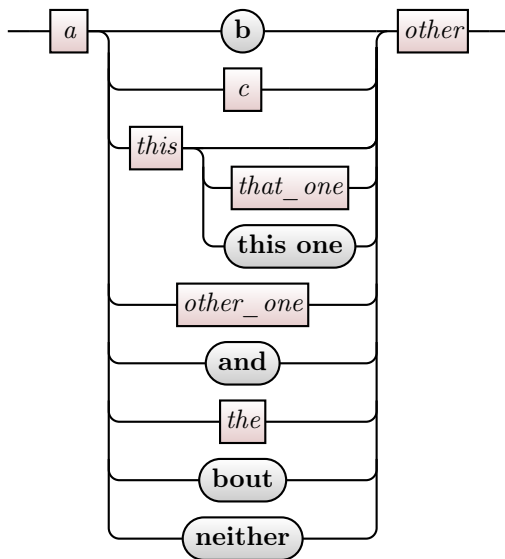
listk



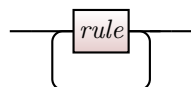
hmm



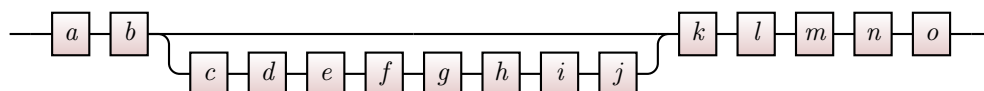
aawesome_list



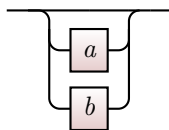
grammar



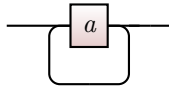
other



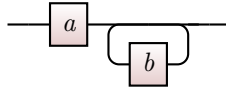
easyone



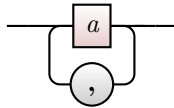
a_better_loop



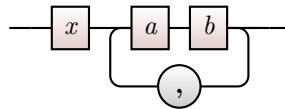
a_better_loopb



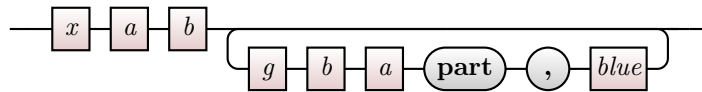
a_much_better_loop



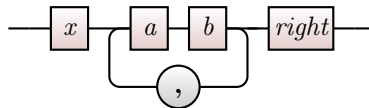
a_much_better_loop



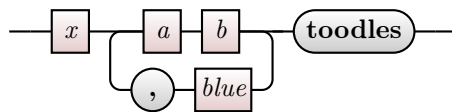
wowzer



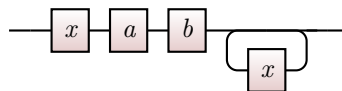
gleep



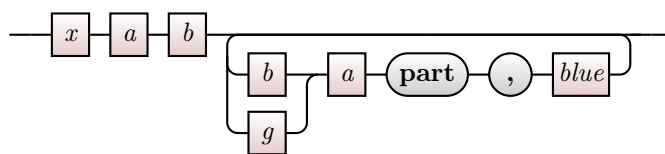
muckrake



broken



scooby



muckrake

