Test reading tree

Bruno Crotman

01/05/2020

Introduction

This is a test, I'm trying to read the abstract syntax tree of a code using PMD rules

```
shell(r"(C:\doutorado\AnaliseTwitter4j\pmd\bin/pmd.bat -d C:\doutorado\AnaliseTwitter4j\match_algorithm
```

```
## Warning in shell("C:\\doutorado\\AnaliseTwitter4j\\pmd\\bin/pmd.bat -
## d C:\\doutorado\\AnaliseTwitter4j\\match_algorithm_description\
## \little-tree -f xml -R C:\\doutorado\\AnaliseTwitter4j\
## \match_algorithm_description\\blockrules\\blockrules.xml -reportfile
## C:\\doutorado\\AnaliseTwitter4j\\match_algorithm_description\
## \oldblock.xml"): 'C:\doutorado\AnaliseTwitter4j\pmd\bin/pmd.bat -d C:
## \doutorado\AnaliseTwitter4j\match_algorithm_description\little-tree -f xml -R C:
## \doutorado\AnaliseTwitter4j\match_algorithm_description\blockrules\blockrules.xml
## -reportfile C:
## \doutorado\AnaliseTwitter4j\match_algorithm_description\oldblock.xml' execution
## failed with error code 4
```

Introduction

```
map_rule_small <- tribble(

    ~rule,                                  ~small_rule,
    "class_or_interface_body",              "class_body",
    "class_or_interface_declaration",       "class_decl",
    "class_or_interface_type",              "class_type",
    "compilation_unit",                     "unit",
    "extends_list",                         "extends",
    "implements_list",                      "implements",
    "import_declaration",                   "import",
    "method",                               "method",
    "name",                                 "name",
    "package",                              "package",
    "type_declaration",                     "type_decl",
    "constructor_declaration",              "constructor",
    "field_declaration",                    "field"

)


alerts <- read_pmd_xml("oldblock.xml") %>%
```

```r
    replace_na(
        list(
            method = "No method"
        )
    ) %>%
    left_join(
        map_rule_small,
        by = c("rule")
    ) %>%
    mutate(
        name = str_glue("{id_alert}:{small_rule}")
    )

max_column <- max(alerts$endcolumn)

alerts_from <- alerts %>%  rename_all(.funs = ~str_glue("{.x}_from"))

alerts_to <- alerts %>%  rename_all(.funs = ~str_glue("{.x}_to"))

all_edges <- alerts_from %>%
    crossing(alerts_to) %>%
    mutate(
        location_begin_from = beginline_from * max_column + begincolumn_from,
        location_begin_to = beginline_to * max_column + begincolumn_to,
        location_end_from = endline_from * max_column + endcolumn_from,
        location_end_to = endline_to * max_column + endcolumn_to
    ) %>%
    filter(id_alert_from != id_alert_to) %>%
    filter(
        location_begin_from <= location_begin_to & location_end_from >= location_end_to
    ) %>%
    select(
        from = id_alert_from,
        to =id_alert_to
    )


descendents <- all_edges %>%
    group_by(from) %>%
    summarise(n_descendents = n())

alerts_sorted <- alerts %>%
    left_join(
        descendents,
        by = c("id_alert" = "from")
    ) %>%
    replace_na(
        list(n_descendents = 0 )
    ) %>%
    arrange(
        desc(n_descendents)
    ) %>%
    mutate(
        id_alert_old = id_alert,
        id_alert = row_number()
    )

map_new_id_alert <- alerts_sorted %>%
    select(
        id_alert_old,
        id_alert
    )

all_edges_new_id <-  all_edges %>%
    left_join(
        map_new_id_alert,
        c("from" = "id_alert_old")
    ) %>%
    mutate(
        from = id_alert
    ) %>%
    select(-id_alert) %>%
    left_join(
        map_new_id_alert,
        c("to" = "id_alert_old")
    ) %>%
    mutate(
        to = id_alert
    ) %>%
    select(-id_alert)

complete_graph <- create_empty(n = 0, directed = TRUE) %>%
    bind_nodes(alerts_sorted ) %>%
    bind_edges(all_edges_new_id)
```

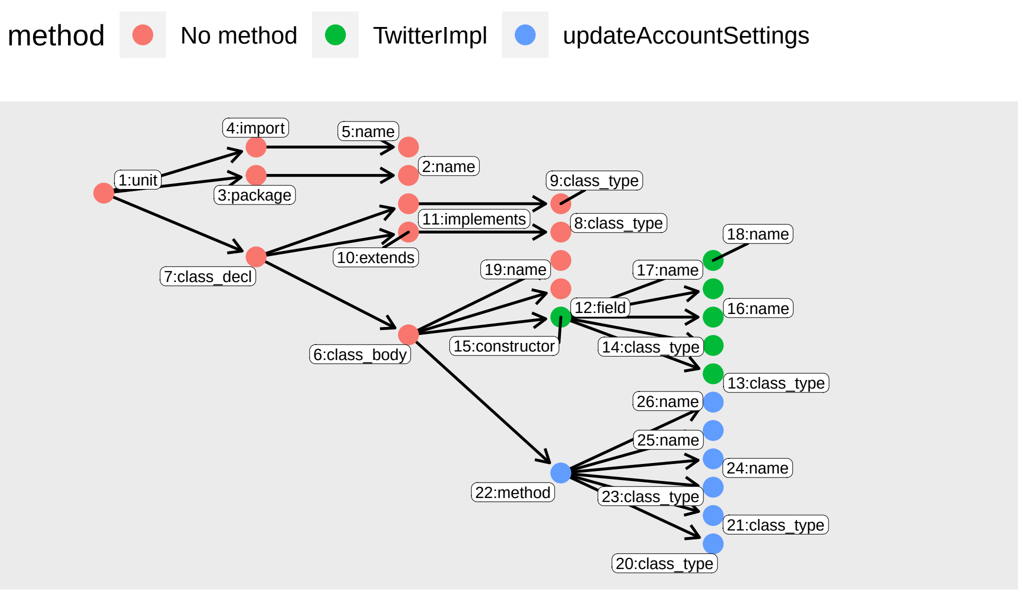
```
## Warning in bind_rows_(x, .id): Vectorizing 'glue' elements may not preserve
## their attributes
```

```r
graph_dfs_tree <- complete_graph %>%
    convert(to_dfs_tree , root = 1, mode = "out" )
```

```
## Warning: `as_quosure()` requires an explicit environment as of rlang 0.3.0.
## Please supply `env`.
## This warning is displayed once per session.
```

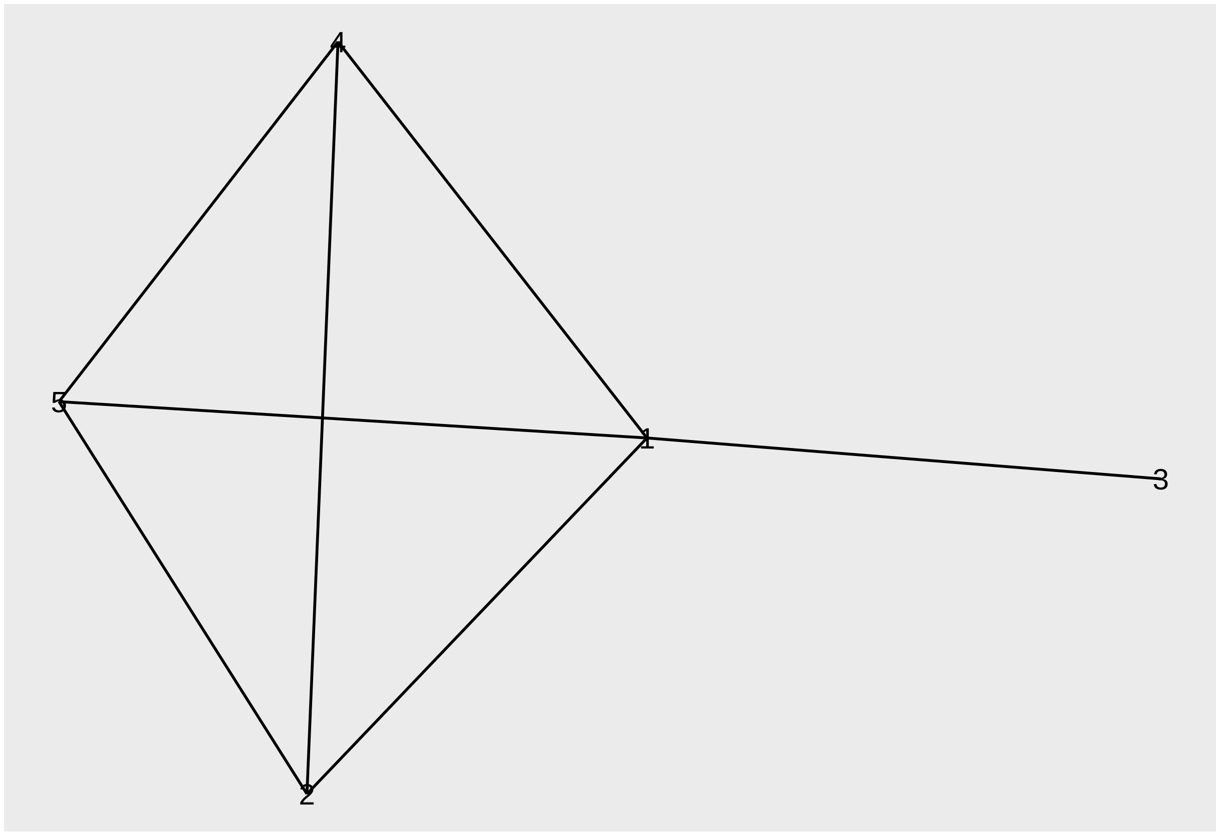
```r
edges <- graph_dfs_tree %>%
    activate(edges)


ggraph(graph_dfs_tree ) +
    geom_edge_link(arrow = arrow(length = unit(2, 'mm')),
                   end_cap = circle(2, 'mm')) +
    geom_node_point(
        aes(color = method),
        size = 3
    ) +
    geom_node_label(
        aes(label = name),
        label.size = 0.1,
        repel = TRUE,
        size = 2,
        label.padding = 0.1
    ) +
    coord_flip() +
    scale_x_reverse(expand =c(-1.2,1.2)) +
    scale_y_continuous(expand =c(-2,2)) +
    theme(
        aspect.ratio = 0.4  ,
        legend.position = "top"
    )
```

```
## Using `tree` as default layout
```

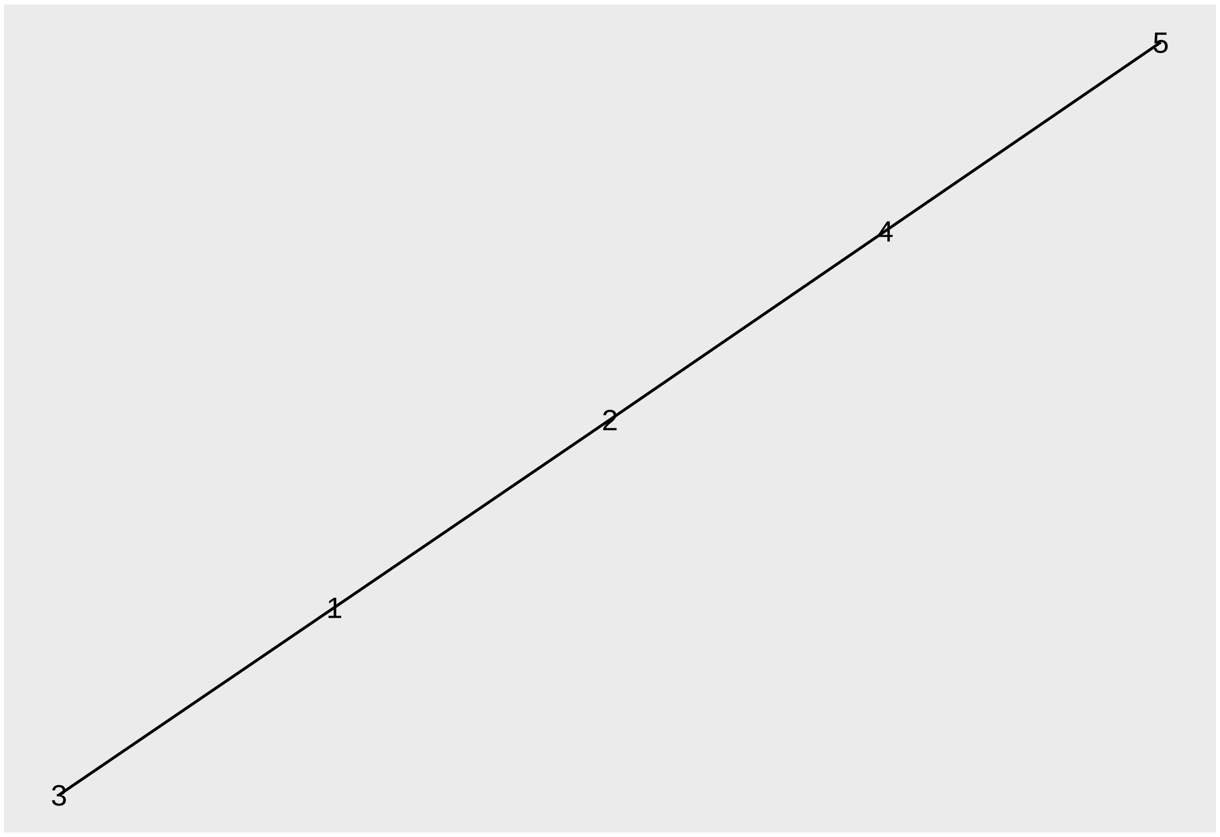


```
nos <- tibble(no = c(1, 2, 3 ,4, 5 ))


grafo <- create_empty(0) %>%
    bind_nodes(nos) %>%
    bind_edges(

        tribble(
            ~from,  ~to,
            1,      2,
            1,      3,
            1,      4,
            2,      4,
            1,      5,
            4,      5,
            2,      5
        )


    )


ggraph(grafo) +
    geom_edge_link() +
    geom_node_text(aes(label = no))
```

```
## Using `stress` as default layout
```



```
arvore <- grafo %>%
    convert(to_dfs_tree, root = 1, mode = "out" )

ggraph(arvore, layout = "kk") +
    geom_edge_link() +
    geom_node_text(aes(label = no))
```



```
arvore %>%
    activate(edges)
```

```
## # A tbl_graph: 5 nodes and 4 edges
## #
## # A rooted tree
## #
## # Edge Data: 4 x 2 (active)
##    from    to
##   <int> <int>
## 1     1     2
## 2     1     3
## 3     2     4
## 4     4     5
## #
## # Node Data: 5 x 2
##      no .tidygraph_node_index
##   <dbl>                 <int>
## 1     1                     1
## 2     2                     2
## 3     3                     3
## # ... with 2 more rows
```