TUGAS BESAR 2
IF3170 INTELIGENSI BUATAN

LAPORAN TUGAS

Diajukan sebagai laporan dari tugas besar 2 Inteligensi Buatan IF3170 pada Semester I Tahun Akademik 2021-2022

Oleh kelompok RADWIMPS - AI ni dekiru:

Juan Louis Rombetasik	13519075
Aria Bachrul Ulum Berlian	13519115
Azmi Muhammad Syazwana	13519151



13519075, 13519115, 13519151

Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung

2021

1. Source

```
;=======================================================
; Author 1: Juan Louis Rombetasik 13519075
; Author 2: Aria Bachrul Ulum Berlian 13519115
; Author 3: Azmi Muhammad syazwana 13519151
; File: breastcancer.clp
; Date: 11/11/2021
; Description: Breast cancer Diagnostic Program
;=======================================================

(deffacts mulai
    (amcp)
)
; ===================== root ====================
(defrule ask-mean-concave-points
    ?val <- (amcp)
    =>
    (retract ?val)
    (printout t "Mean concave points? ")
    (assert (concave-points-input (read)))
)

(defrule mean-concave-points-baik
    ?valid <- (concave-points-input ?num&:(numberp ?num))
    =>
    (retract ?valid)
    (assert (concave-points ?num))
)

(defrule mean-concave-points-buruk
    ?valid <- (concave-points-input ?num&~:(numberp ?num))
    =>
    (retract ?valid)
    (printout t "Mohon masukkan nilai yang valid!" crlf)
    (assert (amcp))
)


; ===================== degree 1 ====================
(defrule ask-worst-radius
    (concave-points ?num&:(<= ?num 0.05))
    =>
    (printout t "Worst radius? ")
```

```
    (assert (worst-radius-input (read)))
)

(defrule worst-radius-baik
    ?valid <- (worst-radius-input ?num&:(numberp ?num))
    =>
    (retract ?valid)
    (assert (worst-radius ?num))
)

(defrule worst-radius-buruk
    ?valid <- (worst-radius-input ?num&~:(numberp ?num))
    ?val <- (concave-points ?cp)
    =>
    (retract ?val ?valid)
    (printout t "Mohon masukkan nilai yang valid!" crlf)
    (assert (concave-points ?cp))
)

(defrule ask-worst-perimeter
    (concave-points ?num&:(> ?num 0.05))
    =>
    (printout t "Worst perimeter? ")
    (assert (worst-perimeter-input (read)))
)

(defrule worst-perimeter-baik
    ?valid <- (worst-perimeter-input ?num&:(numberp ?num))
    =>
    (retract ?valid)
    (assert (worst-perimeter ?num))
)

(defrule worst-perimeter-buruk
    ?valid <- (worst-perimeter-input ?num&~:(numberp ?num))
    ?val <- (concave-points ?cp)
    =>
    (retract ?val ?valid)
    (printout t "Mohon masukkan nilai yang valid!" crlf)
    (assert (concave-points ?cp))
)


; ===================== degree 2 =====================
```

```
(defrule ask-radius-error
    (worst-radius ?num&:(<= ?num 16.83))
    =>
    (printout t "Radius error? ")
    (assert (radius-error-input (read)))
)

(defrule radius-error-baik
    ?valid <- (radius-error-input ?num&:(numberp ?num))
    =>
    (retract ?valid)
    (assert (radius-error ?num))
)

(defrule radius-error-buruk
    ?valid <- (radius-error-input ?num&~:(numberp ?num))
    ?val <- (worst-radius ?wr)
    =>
    (retract ?val ?valid)
    (printout t "Mohon masukkan nilai yang valid!" crlf)
    (assert (worst-radius ?wr))
)

(defrule ask-mean-texture
    (worst-radius ?num&:(> ?num 16.83))
    =>
    (printout t "Mean texture? ")
    (assert (mean-texture-input (read)))
)

(defrule mean-texture-baik
    ?valid <- (mean-texture-input ?num&:(numberp ?num))
    =>
    (retract ?valid)
    (assert (mean-texture ?num))
)

(defrule mean-texture-buruk
    ?valid <- (mean-texture-input ?num&~:(numberp ?num))
    ?val <- (worst-radius ?wr)
    =>
    (retract ?val ?valid)
    (printout t "Mohon masukkan nilai yang valid!" crlf)
    (assert (worst-radius ?wr))
```

```
)


(defrule ask-worst-texture-perimeter
    (worst-perimeter ?num&:(<= ?num 114.45))
    =>
    (printout t "Worst texture? ")
    (assert (worst-texture-input (read)))
)

(defrule worst-texture-perimeter-baik
    ?valid <- (worst-texture-input ?num&:(numberp ?num))
    =>
    (retract ?valid)
    (assert (worst-texture-perimeter ?num))
)

(defrule worst-texture-perimeter-buruk
    ?valid <- (worst-texture-input ?num&~:(numberp ?num))
    ?val <- (worst-perimeter ?wp)
    =>
    (retract ?val ?valid)
    (printout t "Mohon masukkan nilai yang valid!" crlf)
    (assert (worst-perimeter ?wp))
)

(defrule worst-perimeter-over
    (worst-perimeter ?num&:(> ?num 114.45))
    =>
    (assert (hasil 0.0))
)

; ===================== degree 3 =====================
(defrule ask-worst-texture-radius
    (radius-error ?num&:(<= ?num 0.63))
    =>
    (printout t "Worst texture? ")
    (assert (worst-texture-radius-input (read)))
)

(defrule worst-texture-radius-baik
    ?valid <- (worst-texture-radius-input ?num&:(numberp ?num))
    =>
    (retract ?valid)
```

```
    (assert (worst-texture-radius ?num))
)

(defrule worst-texture-radius-buruk
    ?valid <- (worst-texture-input ?num&~:(numberp ?num))
    ?val <- (radius-error ?wp)
    =>
    (retract ?val ?valid)
    (printout t "Mohon masukkan nilai yang valid!" crlf)
    (assert (radius-error ?wp))
)

(defrule ask-mean-smoothness
    (radius-error ?num&:(> ?num 0.63))
    =>
    (printout t "Mean smoothness? ")
    (assert (mean-smoothness-input (read)))
)

(defrule mean-smoothness-baik
    ?valid <- (mean-smoothness-input ?num&:(numberp ?num))
    =>
    (retract ?valid)
    (assert (mean-smoothness ?num))
)

(defrule mean-smoothness-buruk
    ?valid <- (worst-texture-input ?num&~:(numberp ?num))
    ?val <- (radius-error ?wp)
    =>
    (retract ?val ?valid)
    (printout t "Mohon masukkan nilai yang valid!" crlf)
    (assert (radius-error ?wp))
)


(defrule mean-texture-below
    (mean-texture ?num&:(<= ?num 16.19))
    =>
    (assert (hasil 1.0))
)

(defrule ask-concave-points-error
    (mean-texture ?num&:(> ?num 16.19))
```

```
    =>
    (printout t "Concave points error? ")
    (assert (concave-points-error-input (read)))
)

(defrule concave-points-error-baik
    ?valid <- (concave-points-error-input ?num&:(numberp ?num))
    =>
    (retract ?valid)
    (assert (concave-points-error ?num))
)

(defrule concave-points-error-buruk
    ?valid <- (concave-points-error-input ?num&~:(numberp ?num))
    ?val <- (mean-texture ?wr)
    =>
    (retract ?val ?valid)
    (printout t "Mohon masukkan nilai yang valid!" crlf)
    (assert (mean-texture ?wr))
)


(defrule ask-worst-concave-points
    (worst-texture-perimeter ?num&:(<= ?num 25.65))
    =>
    (printout t "Worst concave points? ")
    (assert (worst-concave-points-input (read)))
)

(defrule worst-concave-points-baik
    ?valid <- (worst-concave-points-input ?num&:(numberp ?num))
    =>
    (retract ?valid)
    (assert (worst-concave-points ?num))
)

(defrule worst-concave-points-buruk
    ?valid <- (worst-concave-points-input ?num&~:(numberp ?num))
    ?val <- (worst-texture-perimeter ?wr)
    =>
    (retract ?val ?valid)
    (printout t "Mohon masukkan nilai yang valid!" crlf)
    (assert (worst-texture-perimeter ?wr))
)
```

```
(defrule ask-perimeter-error
    (worst-texture-perimeter ?num&:(> ?num 25.65))
    =>
    (printout t "Perimeter error? ")
    (assert (perimeter-error-input (read)))
)

(defrule perimeter-error-baik
    ?valid <- (perimeter-error-input ?num&:(numberp ?num))
    =>
    (retract ?valid)
    (assert (perimeter-error ?num))
)

(defrule perimeter-error-buruk
    ?valid <- (perimeter-error-input ?num&~:(numberp ?num))
    ?val <- (worst-texture-perimeter ?wr)
    =>
    (retract ?val ?valid)
    (printout t "Mohon masukkan nilai yang valid!" crlf)
    (assert (worst-texture-perimeter ?wr))
)


; ==================== degree 4 ====================
(defrule worst-texture-radius-below
    (worst-texture-radius ?num&:(<= ?num 30.15))
    =>
    (assert (hasil 1.0))
)

(defrule ask-worst-area
    (worst-texture-radius ?num&:(> ?num 30.15))
    =>
    (printout t "Worst area? ")
    (assert (worst-area-input (read)))
)

(defrule worst-area-baik
    ?valid <- (worst-area-input ?num&:(numberp ?num))
    =>
    (retract ?valid)
    (assert (worst-area ?num))
```

```
)

(defrule worst-area-buruk
    ?valid <- (worst-area-input ?num&~:(numberp ?num))
    ?val <- (worst-texture-radius ?wr)
    =>
    (retract ?val ?valid)
    (printout t "Mohon masukkan nilai yang valid!" crlf)
    (assert (worst-texture-radius ?wr))
)


(defrule mean-smoothness-below
    (mean-smoothness ?num&:(<= ?num 0.09))
    =>
    (assert (hasil 1.0))
)
(defrule mean-smoothness-over
    (mean-smoothness ?num&:(> ?num 0.09))
    =>
    (assert (hasil 0.0))
)


(defrule concave-points-error-below
    (concave-points-error ?num&:(<= ?num 0.01))
    =>
    (assert (hasil 0.0))
)
(defrule concave-points-error-over
    (concave-points-error ?num&:(> ?num 0.01))
    =>
    (assert (hasil 1.0))
)


(defrule worst-concave-points-below
    (worst-concave-points ?num&:(<= ?num 0.17))
    =>
    (assert (hasil 1.0))
)
(defrule worst-concave-points-over
    (worst-concave-points ?num&:(> ?num 0.17))
    =>
```

```
    (assert (hasil 0.0))
)


(defrule ask-mean-radius
    (perimeter-error ?num&:(<= ?num 1.56))
    =>
    (printout t "Mean radius? ")
    (assert (mean-radius-input (read)))
)

(defrule mean-radius-baik
    ?valid <- (mean-radius-input ?num&:(numberp ?num))
    =>
    (retract ?valid)
    (assert (mean-radius ?num))
)

(defrule mean-radius-buruk
    ?valid <- (mean-radius-input ?num&~:(numberp ?num))
    ?val <- (perimeter-error ?wr)
    =>
    (retract ?val ?valid)
    (printout t "Mohon masukkan nilai yang valid!" crlf)
    (assert (perimeter-error ?wr))
)

(defrule perimeter-error-over
    (perimeter-error ?num&:(> ?num 1.56))
    =>
    (assert (hasil 0.0))
)

; ====================== degree 5 ======================
(defrule worst-area-below
    (worst-area ?num&:(<= ?num 641.60))
    =>
    (assert (hasil 1.0))
)

(defrule ask-mean-radius-worst-area
    (worst-area ?num&:(> ?num 641.60))
    =>
    (printout t "Mean radius? ")
```

```
    (assert (mean-radius-worst-area-input (read)))
)

(defrule mean-radius-worst-area-baik
    ?valid <- (mean-radius-worst-area-input ?num&:(numberp ?num))
    =>
    (retract ?valid)
    (assert (mean-radius-worst-area ?num))
)

(defrule mean-radius-worst-area-buruk
    ?valid <- (mean-radius-input ?num&~:(numberp ?num))
    ?val <- (worst-area ?wr)
    =>
    (retract ?val ?valid)
    (printout t "Mohon masukkan nilai yang valid!" crlf)
    (assert (worst-area ?wr))
)


(defrule mean-radius-below
    (mean-radius ?num&:(<= ?num 13.34))
    =>
    (assert (hasil 0.0))
)
(defrule mean-radius-over
    (mean-radius ?num&:(> ?num 13.34))
    =>
    (assert (hasil 1.0))
)

; ====================== degree 6 ======================
(defrule ask-mean-texture-mean-radius
    (mean-radius-worst-area ?num&:(<= ?num 13.45))
    =>
    (printout t "Mean texture? ")
    (assert (mean-texture-mean-radius-input (read)))
)

(defrule mean-texture-mean-radius-baik
    ?valid <- (mean-texture-mean-radius-input ?num&:(numberp ?num))
    =>
    (retract ?valid)
    (assert (mean-texture-mean-radius ?num))
```

```
)

(defrule mean-texture-mean-radius-buruk
    ?valid <- (mean-radius-input ?num&~:(numberp ?num))
    ?val <- (mean-radius-worst-area ?wr)
    =>
    (retract ?val ?valid)
    (printout t "Mohon masukkan nilai yang valid!" crlf)
    (assert (mean-radius-worst-area ?wr))
)

(defrule mean-radius-worst-area-over
    (mean-radius-worst-area ?num&:(> ?num 13.45))
    =>
    (assert (hasil 1.0))
)

; ====================== degree 7 ======================
(defrule mean-texture-mean-radius-below
    (mean-texture-mean-radius ?num&:(<= ?num 28.79))
    =>
    (assert (hasil 0.0))
)
(defrule mean-texture-mean-radius-over
    (mean-texture-mean-radius ?num&:(> ?num 28.79))
    =>
    (assert (hasil 1.0))
)
; ================================ Hasil ================================
(defrule kena-kanker
    (hasil 1.0)
    =>
    (printout t "Hasil Prediksi = Terprediksi Kanker Payudara" crlf)
)

(defrule tidak-kena-kanker
    (hasil 0.0)
    =>
    (printout t "Hasil Prediksi = Terprediksi Tidak Kanker Payudara" crlf)
)
```

2. Penjelasan Program, Fakta dan Rule

Strategy *conflict resolution* program: ***Depth Strategy***.

Pembuatan *facts* dan *rules* dari *breast cancer prediction* CLIPS sangatlah sederhana. Pada dasarnya pembuatan rule dari sebuah *node parent* dibuat dengan 3 *rule*, yaitu 1 buah *rule* untuk mendapat *raw input,* dan 2 buah *rules* untuk validasi input (apakah input itu *number* atau input itu bukan *number*). Pembuatan rule dari sebuah *node* daun hanya terdiri satu *rule* yaitu *rule* meng *assert* fakta "hasil". Terdapat total 63 *rules* pada program yang kelompok kami buat. *Rules* tersebut terdiri dari 45 *rules* dari *node* yang bercabang (15 *node parent*), 16 *rules* dari *node* daun, dan sisanya yaitu 2 *rules* untuk *output* hasil prediksi kanker (tidak terkena kanker dan terkena kanker). Penjelasan dari *rule* dan cara *passing* antara rule ada di bawah ini.

Pada sebuah *node parent* terdapat sebuah rule untuk meminta *raw input* (*rule* yang dimulai dengan kata "*ask.")*. Seperti contoh dibawah ini adalah *rule* "*ask-mean-concave-points"* dan *rule* "*ask-worst-radius."* Pada sebuah *node parent* juga terdapat 2 buah rule penyaring untuk memvalidasi (menyaring *raw input*). Penunjuk *rule* penyaring ditandai dengan kata "baik" dan "buruk" di akhir nama sebuah *rule. Raw input* yang "baik" adalah sebuah input dari user yang merupakan sebuah *number* (*float* atau *integer*). Pada *rule* penyaring , *raw input* yang "baik" akan kembali di assert menjadi sebuah fakta yang nantinya digunakan untuk membangkitakan *rule* yang lain. Pada contoh dibawah, *rule "ask-worst-radius"* memerlukan fakta "*concave-points _"* yang merupakan hasil saringan *raw input* "*concave-points-input _."*.Pada *raw input* yang "buruk", akan terjadi penghapusan fakta *raw input* tersebut dan penghapusan fakta *prekondisi* dari *rule* peminta *input* dan meng *assert* kembali prekondisi tersebut untuk membangkitkan *rule* peminta *input* terkait. Seperti contoh di bawah ini, pada *rule "worst-radius-buruk,"* fakta "*concave-points* _" akan dihapus dan di *assert* kembali untuk membangkitkan *rule "ask-worst-radius"* kembali.

```
(deffacts mulai
    (amcp)
)

(defrule ask-mean-concave-points
    ?val <- (amcp)
    =>
    (retract ?val)
    (printout t "Mean concave points? ")
    (assert (concave-points-input (read)))
```

```
)

;VALIDASI RAW INPUT YANG VALID
(defrule mean-concave-points-baik
    ?valid <- (concave-points-input ?num&:(numberp ?num))
    =>
    (retract ?valid)
    (assert (concave-points ?num))
)
;VALIDASI RAW INPUT YANG TIDAK VALID
(defrule mean-concave-points-buruk
    ?valid <- (concave-points-input ?num&~:(numberp ?num))
    =>
    (retract ?valid)
    (printout t "Mohon masukkan nilai yang valid!" crlf)
    (assert (amcp))
)

; ====================== degree 1 ======================
(defrule ask-worst-radius
    (concave-points ?num&:(<= ?num 0.05))
    =>
    (printout t "Worst radius? ")
    (assert (worst-radius-input (read)))
)

(defrule worst-radius-baik
    ?valid <- (worst-radius-input ?num&:(numberp ?num))
    =>
    (retract ?valid)
    (assert (worst-radius ?num))
)

(defrule worst-radius-buruk
    ?valid <- (worst-radius-input ?num&~:(numberp ?num))
    ?val <- (concave-points ?cp)
    =>
    (retract ?val ?valid)
    (printout t "Mohon masukkan nilai yang valid!" crlf)
    (assert (concave-points ?cp))
)
```

Pada sebuah *node* daun, hanya terdapat satu rule, yaitu *rule* untuk meng *assert* fakta "*hasil"* 0 atau 1. Pada kasus dibawah, *rule "mean-radius-below"* akan terpanggil jika hasil saringan *raw input,* fakta *"mean-radius ?num",* memiliki *?num* kurang dari atau sama dengan 13.34. *Rule* ini akan meng *assert* fakta "hasil 0.0."

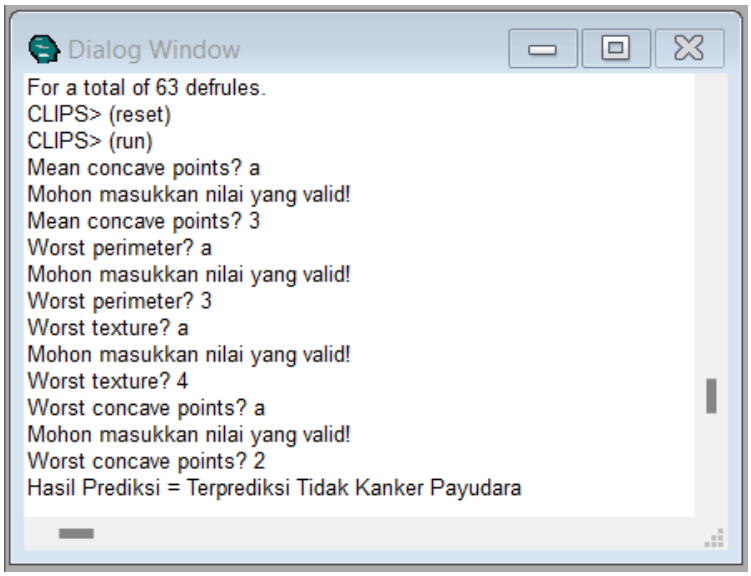
```
(defrule mean-radius-below
    (mean-radius ?num&:(<= ?num 13.34))
    =>
    (assert (hasil 0.0))
)
(defrule mean-radius-over
    (mean-radius ?num&:(> ?num 13.34))
    =>
    (assert (hasil 1.0))
)
```

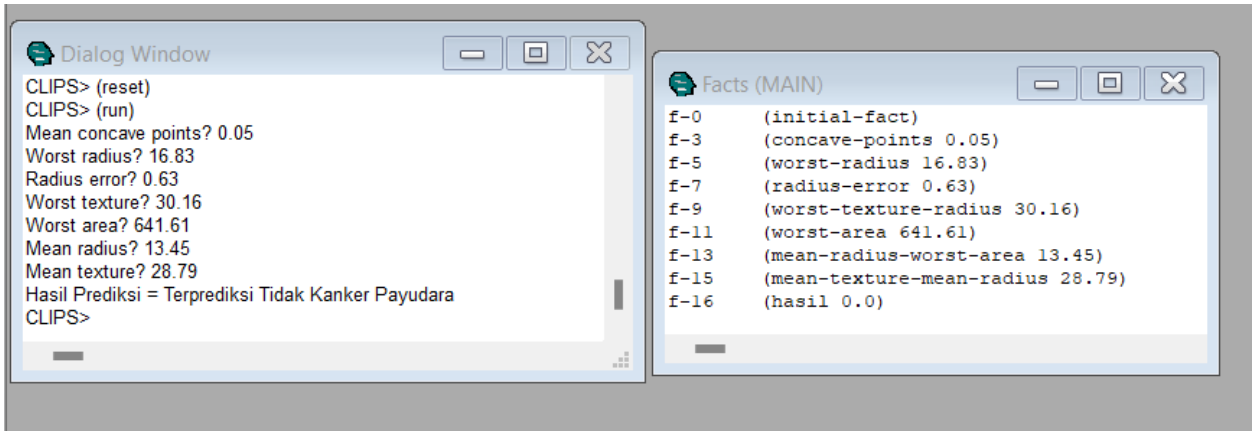
Dan yang terakhir yaitu 2 buah *rule* untuk meng *output* hasil prediksi kanker.

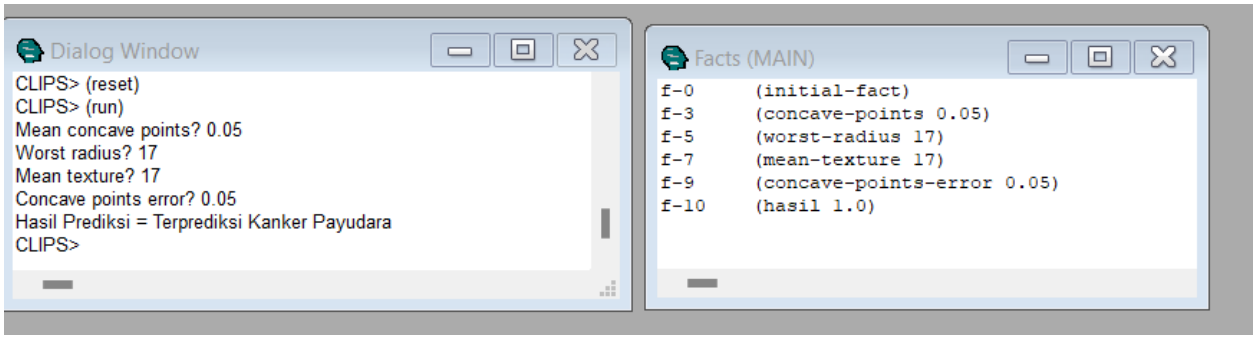
```
; ============================== Hasil ================================
(defrule kena-kanker
    (hasil 1.0)
    =>
    (printout t "Hasil Prediksi = Terprediksi Kanker Payudara" crlf)
)

(defrule tidak-kena-kanker
    (hasil 0.0)
    =>
    (printout t "Hasil Prediksi = Terprediksi Tidak Kanker Payudara" crlf)
)
```

Fakta "hasil *?num*" yang diperoleh dari dari *node* daun akan membangkitkan salah satu dari kedua *rule* diatas. Jika *?num* = 1.0, maka hasil prediksi adalah terkena kanker payudara dan jika *?num* = 0.0, maka hasil prediksi adalah tidak terkena kanker payudara.

Screenshot Program (CLIPS 6.3.x)







Pembagian Tugas

NIM/Nama	Tugas
13519075/Juan Louis R.	*Logic program*, *templating*, laporan
13519115/Aria B. U. Berlian	Implementasi *rule*
13519151/Azmi M. Syazwana	Implementasi *rule*

External

https://github.com/mizuday/AI-ni-dekiru
https://drive.google.com/file/d/1TGbldaNLr0SrshKdNfl1ta0et5VadjET/view?usp=sharing