

**ИНСТИТУТ ЗА МАТЕМАТИКУ И ИНФОРМАТИКУ**

**ПРИРОДНО-МАТЕМАТИЧКИ ФАКУЛТЕТ**

**УНИВЕРЗИТЕТА У КРАГУЈЕВЦУ**

**СЕМИНАРСКИ РАД**

**Предмет: Логичко и функцијско програмирање**

**Тема: Икс Окс у програмском језику Haskell**

**Студент:**

**Душан Стевановић 121/2018**

**Професор:**

**др Татјана Стојановић**

Крагујевац, 2021. год.

Садржај

[**Тема** 3](#_Toc83924796)

[**Опис појединих функција** 3](#_Toc83924797)

[Постављање структуре игре 3](#_Toc83924798)

[Приказ игре у конзоли 4](#_Toc83924799)

[Провера да ли је игра завршена 4](#_Toc83924800)

[Смена стања и унос потеза 6](#_Toc83924801)

[Одређивање оптималног потеза 7](#_Toc83924802)

[Покретање и ток игре 11](#_Toc83924803)

# **Тема**

Тема семинарског рада јесте да се направи непобедиви противник у познатој игри „Икс Окс“ (eng. Tic Tac Toe) у програмском језику haskell. Проблеми које је требало решити су:

* Представљање табле за игру у конзоли којом се представља тренутно стање игре.
* Унос корисничког потеза.
* Израчунавање оптималног потеза рачунара.
* Промена стања игре након сваког потеза.
* Провера да ли је неко од играча победио или је нерешено.

# **Опис појединих функција**

## Постављање структуре игре



Почетно стање игре постављамо у променљивој initialGame, која је типа Game. Тип Game чува таблу за игру (gameBoard :: Board) у облику матрице чији елементи су типа Cell, играча који је тренутно на потезу (gamePlayer :: Player) и информацију о томе да ли је игра у току или је игра завршена и ко је победник (gameState :: State). Оваква структура у haskell-у се назива “record”. Тип Cell чува податак о томе да ли је неко од играча Икс (PlayerX) или Окс (PlayerO) заузео поље или је поље празно (Nothing).

## Приказ игре у конзоли



Функција printBoard уз помоћне функције приказује тренутно стање игре на табли.

## Провера да ли је игра завршена





Скуп ових функција је коришћен за проверу да ли партија доведена до краја и да ли треба наставити са игром. Опис појединих функција:

**isRowFull, isRowfull2** - проверавају да ли прослеђени ред, врста или дијагонала попуњена са три иста симбола и да ли није празан и на основу тога враћају стање GameOver (Just PlayerX) или GameOver (JustPlayerO) уколико листа јесте попуњена или стање Running уколико није.

**getDiagonals, getColumns** – за прослећену таблу издваја колоне и дијагонале које се проверавају у поменутим функцијама, док редове већ имамо јер је сама табла за игру сачувана у облику листи редова.

**stateOR** – како нам претходне фунцкије враћају само стање у појединим деловима табле (редовима, колонама и дијагоналама), потребно добити стање целокупне табле. Ова функција подсећа на логичку операцију OR стим што је true замењено са GameOver стањем а false са Running. Тако да примењена на листу стања редова, колона и дијагоналаона враћа једно резултујуће стање које се односи на целу таблу што је и урађено у наредној функцији.

**checkBoard** – користећи претходне функције, ова функција враћа целокупно стање табле и враћа ко је победио, или да је игра још у току.

**winner** – главна провера се врши у функцији checkBoard али њој фали још провера да ли је игра нерешена па фунцкија winner комплетира одређивање стања игре на основу прослеђене табле.

**isNothing и isFull**– помоћне функције за проверу да ли је поље на табли празно и да ли је цела табла попуњена.

## Смена стања и унос потеза





Како би игра текла од корисника се тражи да унесе свој потез у конзолу и на основу и тако промени стање игре. Функције одговорне за то су:

**getChords** – захтева од корисника да унесе врсту и колону поља на које жели да одигра потез и враћа их у пару као тапл (eng. Tuple).

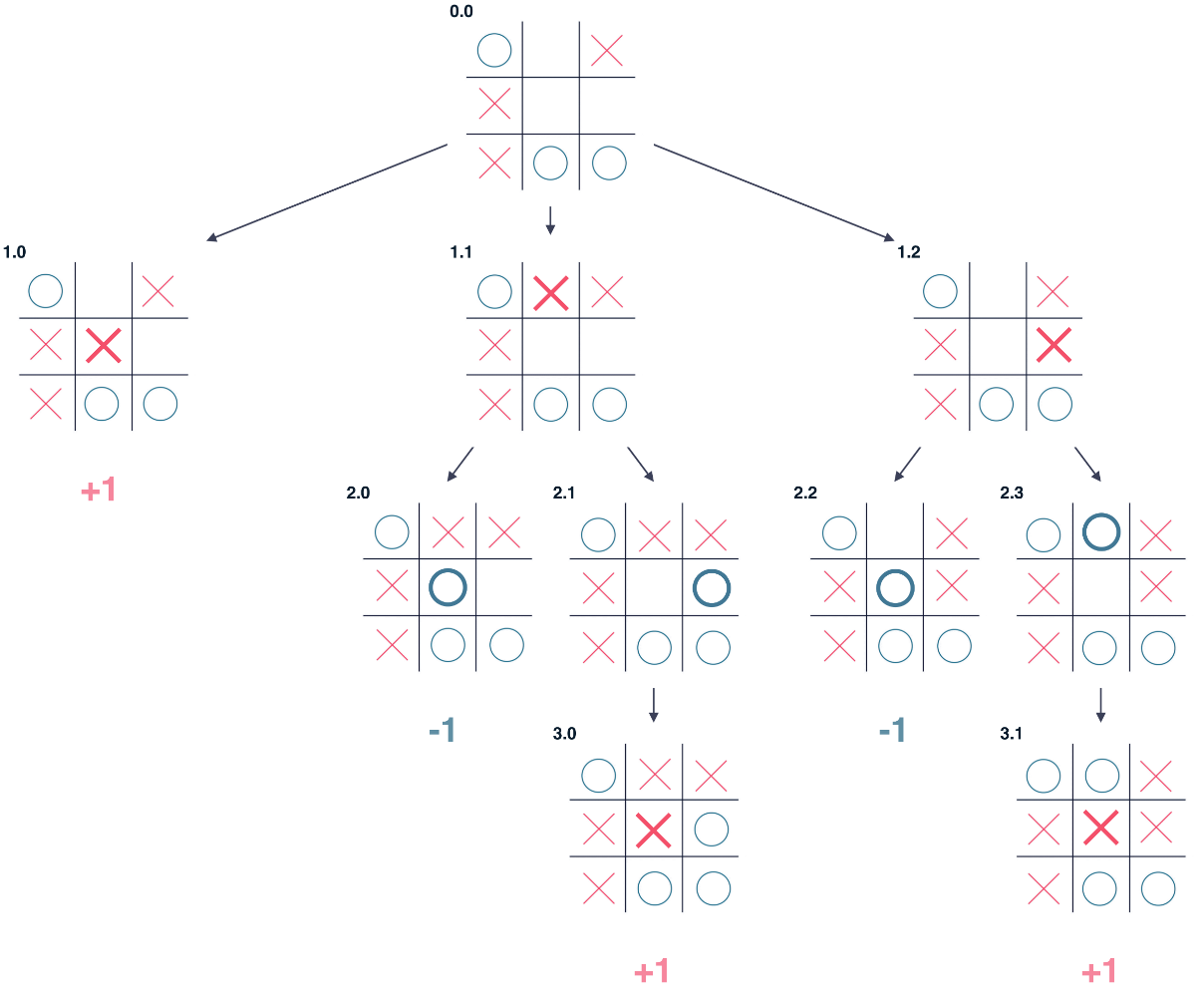
**replaceElem, switchPlaye** – функције за промену стања игре. Прва мења једно поље у листи, служи да попуни поље на табли када играч одигра потез, док друга наизменично мења играче.

**inRange, isCorect** – провера да ли је потез у границама табле за игру.

**playerTurn** – користећи претходне функције проверава валидност потеза и мења стање игре. Ако потез није валидан играч има прилику да понови унос потеза.

## Одређивање оптималног потеза

Један од важнијих делова овог семинарског рада јесте био да се направи алгоритам за рачунање оптималног потеза противника. Задатак је био направити противника који ће своје потезе играти "паметно". За израчунавање оптималног потеза коришћен је Минимакс (eng. Minimax) алгоритам. Минимакс је алгоритам који рекурзивно узима у обзир све могуће исходе игре и на основу тога бира најбољи потез. Као што му име говори циљ минимакс алгоритама је да минимизује максимални губитак (MINImise MAXimum loss). На слици је приказан пример одрећивања оптималног потеза за играча Икс међу три могућа потеза.



Diagram

Description automatically generated

Потез који доноси победу Икс-у вреди 1, онај који доноси победу Окс-у вреди -1, уколико је нерешено потез вреди 0. Гледано одоздо у овом примеру последње потезе је играо Икс, зато међу њима бирамо оне са максималном вредношћу (у оба потеза 3.0 и 3.1 је победио Икс). Након тога мећу потезима које је одиграо Окс бирамо онај са минималном вредношћу. Како алгоритам узима у обзир да је и противник играо оптимално, минимална вредност мећу овим потезима представља најбољи потез противника (у примеру са слике Окс је противник). И на крају поново бирамо најбољи потез за Икс и то је потез 1.0 у коме је и победио.





Минимакс алгоритам је имплементиран уз помоћ две главне функције minimax и score и некилико помоћних функција:

**minimax, score** – функција minimax прихвата тренутно стање игре и тражи потез са максималном вредношћу. За све могуће потезе рачунамо њихову вредност помоћу функције score. Ова функција проверава да ли је прослеђено стање неко од крајњих и уколико јесте врати одговарајућу вредност за потез. Уколико није рекурзивно се позива функција minimax да одреди вредност оптималног потеза противника. Како фунцкија score рачуна вредност потеза само на основу тренутног играча уведен је префикс минус (-scoreNext) при рачунању вредности противниковог потеза. Тиме постижемо наизменично тражења максимума и минимума за сваки следећи ниво рекурзије.

**maksl** – како у функцији minimax све вредности могућих потеза чувамо у листи ова функција проналази највећу међу њима у облику уређеног пара (највећа вредност, потез). Није било протребно правити и функцију за тражење минималне вредности управо због префикса минус који користимо у функцији score.

**converCoords, filterMove getOpponent** – помоћне фукције редом за одрећивање колоне и врсте на основу потеза, филтрирање могућих потеза и одређивање противничког играча.

## Покретање и ток игре



Остало је још све претходно повезати да ради као целина. Главни ток игре спроводи функција **playGame**. У овој функцији се проверава да ли је игра готова и који је играч на реду. Уколико је победио неко од играча или је нерешено, исписује се одговарајућа порука и игра се завршава. Разликујемо два типа потеза, када игра корисник и када игра рачунар. Уколоко је корисник на потезу од њега се тражи да унесе свој поте. Када одигра свој потез функција **computerTurn** користи минимакс алгоритам да пронаће оптималан потез. Покретање игре врши се позивом функције **startGame.**