БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Кафедра интеллектуальных информационных технологий

**Отчёт**

по лабораторной работе №3

по дисциплине

«Логические основы интеллектуальных систем»

Выполнил студент группы 021702

Локтев К.А..

Проверил

Ивашенко В.П.

Минск 2023

Вариант **9**

**Тема:** логическое программирование поиска решения задачи.

**Цель:** приобрести навыки логического программирования поиска решения задачи.

**Задачи:**

1. Изучить основы логического программирования на языке Prolog
2. Освоить методы поиска решений задач в логическом программировании
3. Написать программу на языке Prolog, способную решать задачу, соответствующую варианту, используя полученные знания и навыки

**Условие задачи:** требуется расставить ферзей на шахматной доске так, чтобы ни один ферзь не находился под боем другого ферзя.

**Входные данные:** исходная расстановка ферзей на шахматной доске.

**Выходные данные:** “истина”, если перемещение ферзей при заданной начальной расстановке прошло успешно (в этом случае конечная расстановка будет выведена в стандартный вывод); иначе – “ложь”.

**Использование результатов:**

* Рассматривались алгоритмы решения задачи об N-ферзях из книги И.Братко “Алгоритмы искусственного интеллекта на языке PROLOG”
* Рассматривалось объеснение и алгоритмы решения задачи об N-ферзях в статье автора “Чулан” под названием “Prolog - снова ферзи!”
* Рассматривалось и принималось за основу решение задачи об N-ферзях в статье анонимного автора на портале pro-prof.com.

**Исходные данные:**

Грамматика  языка  PROLOG.

<ПРОЛОГ-предложение>  ::=  <правило>  |  <факт>  |  <запрос>

<правило>  ::=  <заголовок>  ‘:-’<тело>

<факт>  ::=  <заголовок>  ‘.’

<запрос>  ::=  <тело>‘.’

<тело>  ::=  <цель>  /’,’<цель>/’.’

<заголовок>::=  <предикат>

<цель>::=  <предикат>  |<выражение>

<предикат>::=  <имя>/  ‘(‘<терм>  /’,’<терм>/  ‘)’/

<терм>::=  <атом>  |<предикат>|<список>

<атом>::=  <переменная>  |<число>  |<строка>|<имя>

<список>::=  <список  с  заголовком>|  <простой  список>

<список  с  заголовком  >::=  ‘[‘  <терм  >/’,’<терм>/’|’  <  терм>’]’

<  простой  список>::=  ‘[‘  <терм  >/’,’<терм>/’]’|‘[”]’

<оператор>::=  ‘is’  |  ’=’  |  ‘==’  |  ’¯’  |  ’>=’  |  ’=<’  |  ‘=¯’  |  ‘=:=’

|  |  |
| --- | --- |
| ,- конъюнкция | ;- дизъюнкция |
| :-- импликация | \+- отрицание |
| =:=- эквиваленция | . - конец правила |

**Реализация**

Для реализации был выбран язык Prolog версии 8.4.0.

**Алгоритм:**

1. QueensMoved = 0, CurrState = <введённая пользователем исходная расстановка ферзей>
2. Если ни один ферзь не находится под боем другого ферзя, алгоритм завершает работу со значением “истина”
3. QueensMoved += 1
4. Queen = <ферзь, находящийся в списке CurrState на позиции QueensMoved (индексация начинается с единицы)>
5. Ферзь Queen перемещается на произвольную клетку, не находящуюся под боем ни одного ферзя
6. Алгоритм переходит к пункту 2

**Код программы:**

beats(queen(XF, YF), cell(X, Y)):-

    \+ (X == XF, Y == YF),

    (X == XF; Y == YF; X-Y =:= XF-YF; X+Y =:= XF+YF).

beaten(Cell, Queens):-

    member(Queen, Queens),

    beats(Queen, Cell).

beaten(cell(X, Y), Queens, ignored\_queen(XF, YF)):-

    selectchk(queen(XF, YF), Queens, NewQueens),

    member(Queen, NewQueens),

    beats(Queen, cell(X, Y)).

occupied(cell(X, Y), Queens):-

    member(queen(XF, YF), Queens),

    X == XF,

    Y == YF.

no\_queens\_beat\_each\_other([], \_).

no\_queens\_beat\_each\_other([queen(XF, YF)|RemainingQueens], Queens):-

    \+ beaten(cell(XF, YF), Queens),

    no\_queens\_beat\_each\_other(RemainingQueens, Queens).

move\_queen\_to\_non\_beaten\_cell\_and\_update\_state(queen(XF, YF), CurrState, NewState):-

    \+ beaten(cell(XF, YF), CurrState),

    NewState = CurrState.

move\_queen\_to\_non\_beaten\_cell\_and\_update\_state(queen(XF, YF), CurrState, NewState):-

    beaten(cell(XF, YF), CurrState),

    RowsAndColumns = [1, 2, 3, 4, 5, 6, 7, 8],

    member(X, RowsAndColumns),

    member(Y, RowsAndColumns),

    \+ beaten(cell(X, Y), CurrState, ignored\_queen(XF, YF)),

    \+ occupied(cell(X, Y), CurrState),

    !,

    select(queen(XF, YF), CurrState, queen(X, Y), NewState).

solve(CurrState, \_):-

    no\_queens\_beat\_each\_other(CurrState, CurrState),

    write(CurrState).

solve(CurrState, QueensMoved):-

    \+ no\_queens\_beat\_each\_other(CurrState, CurrState),

    NewQueensMoved is QueensMoved + 1,

    nth1(NewQueensMoved, CurrState, Queen),

    move\_queen\_to\_non\_beaten\_cell\_and\_update\_state(Queen, CurrState, NewState),

    solve(NewState, NewQueensMoved).

**Дерево вывода** представлено на рисунке 1.

**Предикаты**

|  |  |
| --- | --- |
| Название | Обозначение |
| beats | A |
| beaten | B |
| member | C |
| occupied | D |
| solve | E |
| move\_queen\_to\_non\_beaten\_cell\_and\_update\_state | F |
| no\_queens\_beat\_each\_other | G |
| is | H |
| ignored\_queen | I |
| queen | J |
| cell | K |
| == | L |
| \+ | M |
| =:= | N |
| nth1 | O |

**Термы и функциональные термы**

|  |  |
| --- | --- |
| Название | Обозначение |
| NewQueensMoved | I |
| QueensMoved | J |
| RowsAndColumns | K |
| RemainingQueens | L |
| Cell | M |
| Queens | N |
| Queen | O |
| NewQueens | P |
| CurrState | R |
| NewState | s |
| X | X |
| Y | Y |
| XF | XF |
| YF | YF |
| = | t |
| [\_|\_] | u |
| - | v |
| + | w |

**Эквивалентная логическая формула на языке логики предикатов предиката в Prolog:**

∀r ∀j (E(r, j) → ¬(∃r' ∃s' (G(r', s') ∧ ∃i (t(j', j + 1) ∧ O(i, s', o) ∧ ((C(o, s') ∧ ∃X∃Y∃XF∃YF∃k (A(o, K) ∧ k = (1, 2, 3, 4, 5, 6, 7, 8) ∧ (C(XF, k) ∧ (C(YF, k) ∧ ((H(v(X, Y), v(XF, YF)) ∧ ¬w) ∨ ((H(X, XF) ∨ (H(Y, YF) ∨ (H(v(X, Y), v(XF, YF)) ∨ H(w(X, Y), w(XF, YF)))))) ∧ ∃p (u(o, XF, YF, s', p) ∧ ((C(o, p) ∧ ((C(X, p) ∧ (A(o, K) ∧ ¬w)) ∨ t(X, p))) ∨ (D(m(X, Y), p) ∧ ¬w(X, p)))) ∧ ¬w ∧ ∃s (u(o, XF, YF, s', o, s) ∧ E(s, i)))))))) ∨ ((s' = s') ∧ E(s', i)))))) ∨ (∃r ∃s (G(r, r) ∧ P(r) ∧ ¬w))

**Вывод:**

В рамках данной лабораторной работы были изучены основы логического программирования на языке Prolog и освоены методы поиска решений задач в логическом программировании. Была написана программа на языке Prolog, решающая задачу по размещению N-2 ферзей на доске размером N x N так, чтобы они не били друг друга.

В процессе решения задачи были использованы основные конструкции языка Prolog, такие как предикаты, правила, списки и рекурсия. Для решения задачи был использован алгоритм поиска всех возможных комбинаций расположения ферзей на доске с помощью рекурсивной функции.

В ходе работы были приобретены навыки логического программирования, такие как построение логических формул, использование встроенных предикатов и функций, построение правил вывода, реализация алгоритмов поиска решений задач.

В целом, лабораторная работа позволила познакомиться с основами логического программирования на языке Prolog и использовать полученные знания для решения конкретной задачи.

**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

[1] Статья "PROLOG ― снова ферзи!" [Электронный ресурс]. – Режим доступа :https://habr.com/ru/sandbox/2621/. – Дата доступа: 05.05.2023

[2] Статья "Задача об N ферзях SWI Prolog" [Электронный ресурс]. – Режим доступа :https://pro-prof.com/forums/topic/%D0%B7%D0%B0%D0%B4%D0%B0%D1%87%D0%B0-%D0%BE-8-%D1%84%D0%B5%D1%80%D0%B7%D1%8F%D1%85-swi-prolog. – Дата доступа: 05.05.2023

[3] [ВЫСШАЯ АТТЕСТАЦИОННАЯ КОМИССИЯ РЕСПУБЛИКИ БЕЛАРУСЬ](https://vak.gov.by/) [Электронный ресурс]. – Режим доступа: <https://vak.gov.by/bibliographicDescription>. – Дата доступа: 27.03.2023.

[4] ГОСТ 19.701-90 [Электронный ресурс]. – Режим доступа : [https://www.-google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&cad=rja&uact=8&ved=2ahUKEwin6qiTibT9AhXJ5KQKHTCEDZ0QFnoECBEQAQ&url=http%3A%2F%2Fwww.pntd.ru%2F19.701.htm&usg=AOvVaw1Kw-PMg6SnrnpyzJ-XR4fbS](about:blank). – Дата доступа: 27.03.2023.

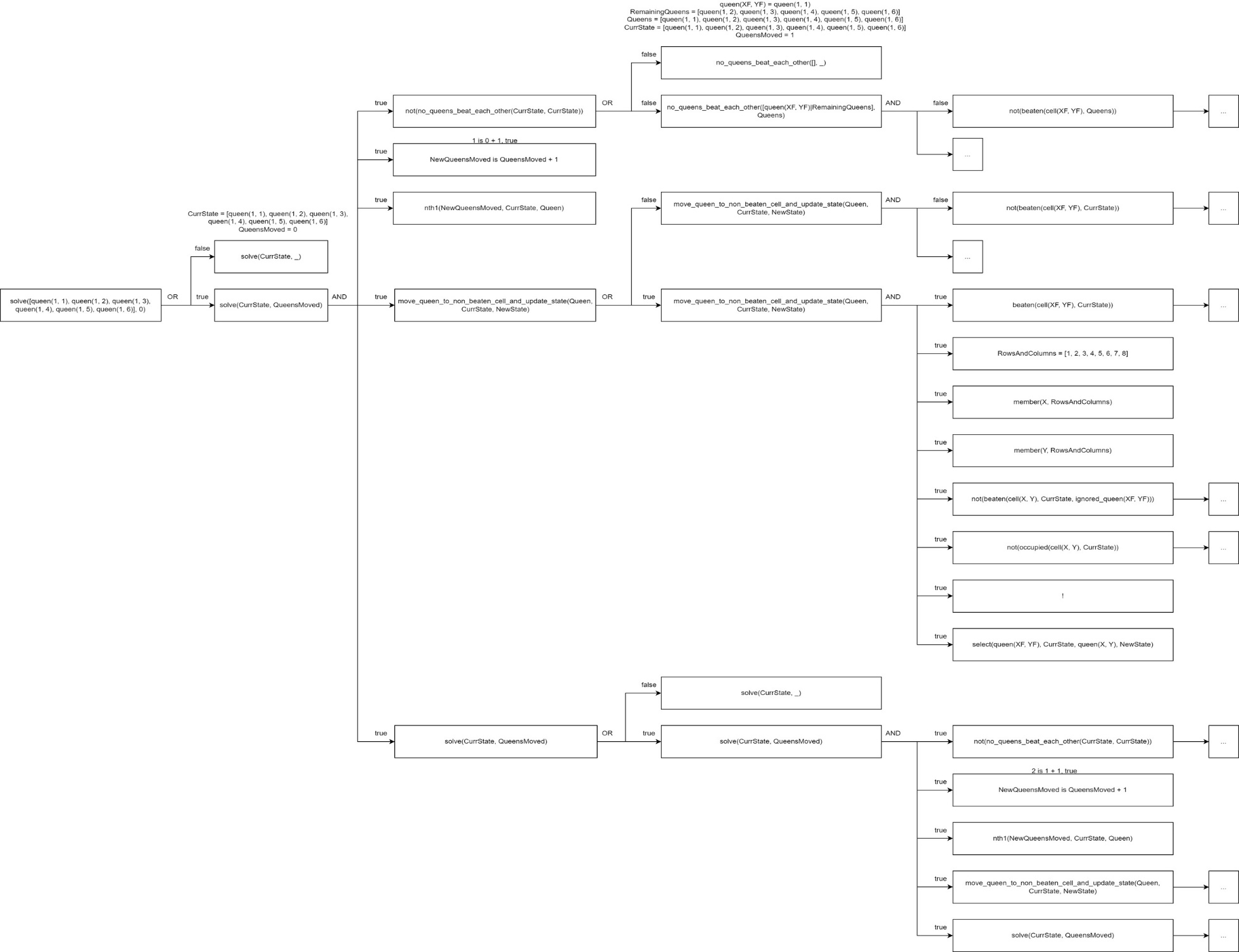


Рисунок 1. Дерево решений программы