

ТИТУЛ

Цель работы

Практическое закрепление теоретических основ информированного (эвристического) поиска с использованием продукционного программирования в среде CLIPS.

Задачи

1. Рассмотреть принципы эвристического поиска в среде продукционного программирования CLIPS на конкретном примере – головоломка 8-ка. Рассматриваем два вида эвристических функций h_1 и h_2 :
 - h_1 – число фишек, стоящих не на своем месте;
 - h_2 – суммарное по всем фишкам число шагов до целевого положения (манхэттенское расстояние).
2. Реализовать в программе подсчет временной и емкостной сложности и сравнить с результатами реализации на императивном языке программирования.

Вариант 1:

Начальное состояние:

5	8	3
4		2
7	6	1

Конечное (целевое) состояние:

1	2	3
4	5	6
7	8	

Описание выбранных структур данных, представления функции определения последователей

Для представления узлы был создан шаблон Node. Узел представляется индификатором (id), 9 слотами для положения фишек (состоянием) (LU, CU, RU, LM, CM, RM, LD, CD, RD), стоимостью (глубиной) (g), статусом (status), индификатором родителя (parent) и значением целевой функции (f):

(deftemplate Node

(slot id(type NUMBER) (default 0)) ; индификатор

(slot LU (type NUMBER)) ; left up

(slot CU (type NUMBER)) ; center up

(slot RU (type NUMBER)) ; right up

(slot LM (type NUMBER)) ; left mid

(slot CM (type NUMBER)) ; center mid

(slot RM (type NUMBER)) ; right mid

(slot LD (type NUMBER)) ; left down

(slot CD (type NUMBER)) ; center down

(slot RD (type NUMBER)) ; right down

(slot g (type NUMBER)) ; cost, depth

(slot status(type NUMBER) (default 0)) ; статус вершины: 0 – не раскрыта, 1 – раскрыта, 2 – соответствует решению

(slot parent (type NUMBER)) ; id родителя

(slot f (type NUMBER)) ; значение целевой функции для данной вершины $f = g + h$

);

Начальные и конечные состояния были определены как глобальные переменные:

```
(defglobal
  ?*HX* = 2 ; 1 if h1 or 2 if h2
  ?*DEBUG* = 0 ; 1 if пошаговый режим, 0 if сквозной режим
  ?*ID* = 0
  ?*NODE_COUNT* = 1 ; 1, потому что рутовый есть изначально
  ?*ITER_COUNT* = 0
  ?*init_LU* = 5
  ?*init_CU* = 8
  ?*init_RU* = 3
  ?*init_LM* = 4
  ?*init_CM* = 0
  ?*init_RM* = 2
  ?*init_LD* = 7
  ?*init_CD* = 6
  ?*init_RD* = 1
  ?*goal_LU* = 1
  ?*goal_CU* = 2
  ?*goal_RU* = 3
  ?*goal_LM* = 4
  ?*goal_CM* = 5
  ?*goal_RM* = 6
  ?*goal_LD* = 7
  ?*goal_CD* = 8
  ?*goal_RD* = 0
);
```

Для определения достижения конечного состояния было определено правило test_goal:

```
(defrule test_goal
  (declare (salience 500))
  ?f_addr <- (Node (id ?v_id) (LU ?v_LU) (CU ?v_CU) (RU ?v_RU)
    (LM ?v_LM) (CM ?v_CM) (RM ?v_RM)
    (LD ?v_LD) (CD ?v_CD) (RD ?v_RD)
    (g ?v_g) (status ~2) (parent ?v_parent) (f ?v_f))
```

```

    )
    (test (= ?v_LU ?*goal_LU*));
    (test (= ?v_CU ?*goal_CU*));
    (test (= ?v_RU ?*goal_RU*));
    (test (= ?v_LM ?*goal_LM*));
    (test (= ?v_CM ?*goal_CM*));
    (test (= ?v_RM ?*goal_RM*));
    (test (= ?v_LD ?*goal_LD*));
    (test (= ?v_CD ?*goal_CD*));
    (test (= ?v_RD ?*goal_RD*));
    =>
    (modify ?f_addr(status 2));
    (printout t crlf "=====" crlf
    "id=" ?v_id " g=" ?v_g " f=" ?v_f " parent=" ?v_parent crlf
    ?v_LU ?v_CU ?v_RU crlf
    ?v_LM ?v_CM ?v_RM crlf
    ?v_LD ?v_CD ?v_RD crlf);
    );

```

Эвристическая функция h1 возвращает кол-во “плиток” (“фишек”) не на своём месте.

```

(deffunction h1( ?cur_LU ?cur_CU ?cur_RU
    ?cur_LM ?cur_CM ?cur_RM
    ?cur_LD ?cur_CD ?cur_RD)
    (bind ?a 0);
    (if (not (= ?cur_LU ?*goal_LU*)) then (bind ?a (+ ?a 1)))
    (if (not (= ?cur_CU ?*goal_CU*)) then (bind ?a (+ ?a 1)))
    (if (not (= ?cur_RU ?*goal_RU*)) then (bind ?a (+ ?a 1)))
    (if (not (= ?cur_LM ?*goal_LM*)) then (bind ?a (+ ?a 1)))

```

```

(if (not (= ?cur_CM ?*goal_CM*)) then (bind ?a (+ ?a 1)))
(if (not (= ?cur_RM ?*goal_RM*)) then (bind ?a (+ ?a 1)))
(if (not (= ?cur_LD ?*goal_LD*)) then (bind ?a (+ ?a 1)))
(if (not (= ?cur_CD ?*goal_CD*)) then (bind ?a (+ ?a 1)))
(if (not (= ?cur_RD ?*goal_RD*)) then (bind ?a (+ ?a 1)))
(return ?a);
);

```

Эвристическая функция h2 возвращает сумму Манхэттенских расстояний “плиток” (“фишек”) от из целевой (конечной) позиции.

```

(deffunction manhattan(?v ?i ?j)
(if (= ?v ?*goal_LU*) then (bind ?i_g 0));
(if (= ?v ?*goal_LU*) then (bind ?j_g 0));
(if (= ?v ?*goal_CU*) then (bind ?i_g 0));
(if (= ?v ?*goal_CU*) then (bind ?j_g 1));
(if (= ?v ?*goal_RU*) then (bind ?i_g 0));
(if (= ?v ?*goal_RU*) then (bind ?j_g 2));
(if (= ?v ?*goal_LM*) then (bind ?i_g 1));
(if (= ?v ?*goal_LM*) then (bind ?j_g 0));
(if (= ?v ?*goal_CM*) then (bind ?i_g 1));
(if (= ?v ?*goal_CM*) then (bind ?j_g 1));
(if (= ?v ?*goal_RM*) then (bind ?i_g 1));
(if (= ?v ?*goal_RM*) then (bind ?j_g 2));
(if (= ?v ?*goal_LD*) then (bind ?i_g 2));
(if (= ?v ?*goal_LD*) then (bind ?j_g 0));
(if (= ?v ?*goal_CD*) then (bind ?i_g 2));
(if (= ?v ?*goal_CD*) then (bind ?j_g 1));
(if (= ?v ?*goal_RD*) then (bind ?i_g 2));
(if (= ?v ?*goal_RD*) then (bind ?j_g 2));

```

```

(return (+ (abs (- ?i ?i_g)) (abs (- ?j ?j_g))));
)
(deffunction h2(  ?cur_LU ?cur_CU ?cur_RU
                 ?cur_LM ?cur_CM ?cur_RM
                 ?cur_LD ?cur_CD ?cur_RD)
(bind ?a 0);
(bind ?a (+ ?a (manhattan ?cur_LU 0 0)));
(bind ?a (+ ?a (manhattan ?cur_CU 0 1)));
(bind ?a (+ ?a (manhattan ?cur_RU 0 2)));
(bind ?a (+ ?a (manhattan ?cur_LM 1 0)));
(bind ?a (+ ?a (manhattan ?cur_CM 1 1)));
(bind ?a (+ ?a (manhattan ?cur_RM 1 2)));
(bind ?a (+ ?a (manhattan ?cur_LD 2 0)));
(bind ?a (+ ?a (manhattan ?cur_CD 2 1)));
(bind ?a (+ ?a (manhattan ?cur_RD 2 2)));
(return ?a);
);

```

Описание алгоритма

В CLIPS алгоритм реализуется правилами, у которых есть приоритет, поэтому порядок действий обеспечивается приоритетами. Описание эвристического алгоритма поиска в среде продукционного программирования CLIPS:

1. Если существует узел с повторным состоянием, то удалить его, если его целевая функция больше, иначе изменить родителя, стоимость и значение целевой функции изначальному раскрытому узлу (приоритет 1000).
2. Если есть узел, состояние которого совпадает с целевым, то изменить ему статус на 2 (приоритет 500).
3. Если появился узел, статус родителя которого не 2, со статусом 2, то изменить статус его родителя тоже на 2 (приоритет 500).
4. Если появился хотя бы один узел со статусом 2, то удалить все узлы, статус которых не 2 (приоритет 400).
5. Если не осталось узлов со статусом 0 или 2, то закончить работу с сообщением о невозможности найти ответ (приоритет 200).
6. Если есть узел со статусом 2, то закончить работу с сообщением о найденном ответе (приоритет 200).
7. Определение текущего минимум целевой функции (приоритет 150-175).
8. Раскрыть соседние узлы у узла с минимальным значением целевой функции (приоритет 100).

Результаты работы

Исходный код реализации представлен в приложении. В defglobal можно изменить ?*НХ* на 1, чтобы использовать эвристическую функцию h1, или на 2, чтобы использовать эвристическую функцию h2; чтобы переключиться в пошаговый режим, нужно выставить ?*DEBUG* в значение 1.

Примеры протоколов выполнения программ для эвристических функций h1 и h2 в пошаговом и сквозном режимах представлены на рисунках 1, 2, 3 и 4.

```

Dialog Window
Defining defrule: make_new_from_RD =j+j+j
TRUE
CLIPS> (reset)
CLIPS> (run)

=====
id=21144 g=22 f=22 parent=21141
123
456
780
^
id=21141 g=21 f=23 parent=21139
123
456
708
^
id=21139 g=20 f=23 parent=18836
123
456
078
^
id=18836 g=19 f=23 parent=18833
123
056
478
^
id=18833 g=18 f=23 parent=18830
123
506
478
^
id=18830 g=17 f=23 parent=14520
103
526
478
...

Facts (MAIN)
f-0 (initial-fact)
f-3 (if FALSE printout t crlf "Init node: " crlf "id=1 " "g=0 "
f-30618 (min 23)
f-31156 (Node (id 21144) (LU 1) (CU 2) (RU 3) (LM 4) (CM 5) (RM 6) (C
f-31157 (Node (id 21141) (LU 1) (CU 2) (RU 3) (LM 4) (CM 5) (RM 6) (C
f-31158 (Node (id 21139) (LU 1) (CU 2) (RU 3) (LM 4) (CM 5) (RM 6) (C
f-31159 (Node (id 18836) (LU 1) (CU 2) (RU 3) (LM 0) (CM 5) (RM 6) (C
f-31160 (Node (id 18833) (LU 1) (CU 2) (RU 3) (LM 5) (CM 0) (RM 6) (C
f-31161 (Node (id 18830) (LU 1) (CU 0) (RU 3) (LM 5) (CM 2) (RM 6) (C
f-31162 (Node (id 14520) (LU 0) (CU 1) (RU 3) (LM 5) (CM 2) (RM 6) (C
f-31163 (Node (id 7246) (LU 5) (CU 1) (RU 3) (LM 0) (CM 2) (RM 6) (LM
f-31164 (Node (id 3080) (LU 5) (CU 1) (RU 3) (LM 4) (CM 2) (RM 6) (LM
f-31165 (Node (id 1042) (LU 5) (CU 1) (RU 3) (LM 4) (CM 2) (RM 6) (LM
f-31166 (Node (id 1040) (LU 5) (CU 1) (RU 3) (LM 4) (CM 2) (RM 6) (LM
f-31167 (Node (id 683) (LU 5) (CU 1) (RU 3) (LM 4) (CM 2) (RM 0) (LD
f-31168 (Node (id 328) (LU 5) (CU 1) (RU 3) (LM 4) (CM 0) (RM 2) (LD
f-31169 (Node (id 325) (LU 5) (CU 0) (RU 3) (LM 4) (CM 1) (RM 2) (LD
f-31170 (Node (id 197) (LU 5) (CU 3) (RU 0) (LM 4) (CM 1) (RM 2) (LD
f-31171 (Node (id 195) (LU 5) (CU 3) (RU 2) (LM 4) (CM 1) (RM 0) (LD
f-31172 (Node (id 190) (LU 5) (CU 3) (RU 2) (LM 4) (CM 0) (RM 1) (LD
f-31173 (Node (id 66) (LU 5) (CU 3) (RU 2) (LM 4) (CM 8) (RM 1) (LD
f-31174 (Node (id 64) (LU 5) (CU 3) (RU 2) (LM 4) (CM 8) (RM 1) (LD
f-31175 (Node (id 27) (LU 5) (CU 3) (RU 2) (LM 4) (CM 8) (RM 0) (LD
f-31176 (Node (id 18) (LU 5) (CU 3) (RU 0) (LM 4) (CM 8) (RM 2) (LD
f-31177 (Node (id 2) (LU 5) (CU 0) (RU 3) (LM 4) (CM 8) (RM 2) (LD 7
f-31178 (Node (id 1) (LU 5) (CU 8) (RU 3) (LM 4) (CM 0) (RM 2) (LD 7

Dialog Window
^
id=66 g=5 f=12 parent=64
532
481
706
^
id=64 g=4 f=10 parent=27
532
481
760
^
id=27 g=3 f=10 parent=18
532
480
761
^
id=18 g=2 f=9 parent=2
530
482
761
^
id=2 g=1 f=7 parent=1
503
482
761
^
id=1 g=0 f=6 parent=0
583
402
761

Solution found!
Nodes number: 12134
Inter count: 8017
CLIPS> █

```

Рисунок 1 — Пример протокола выполнения программы в сквозном режиме с эвристической функцией h1.

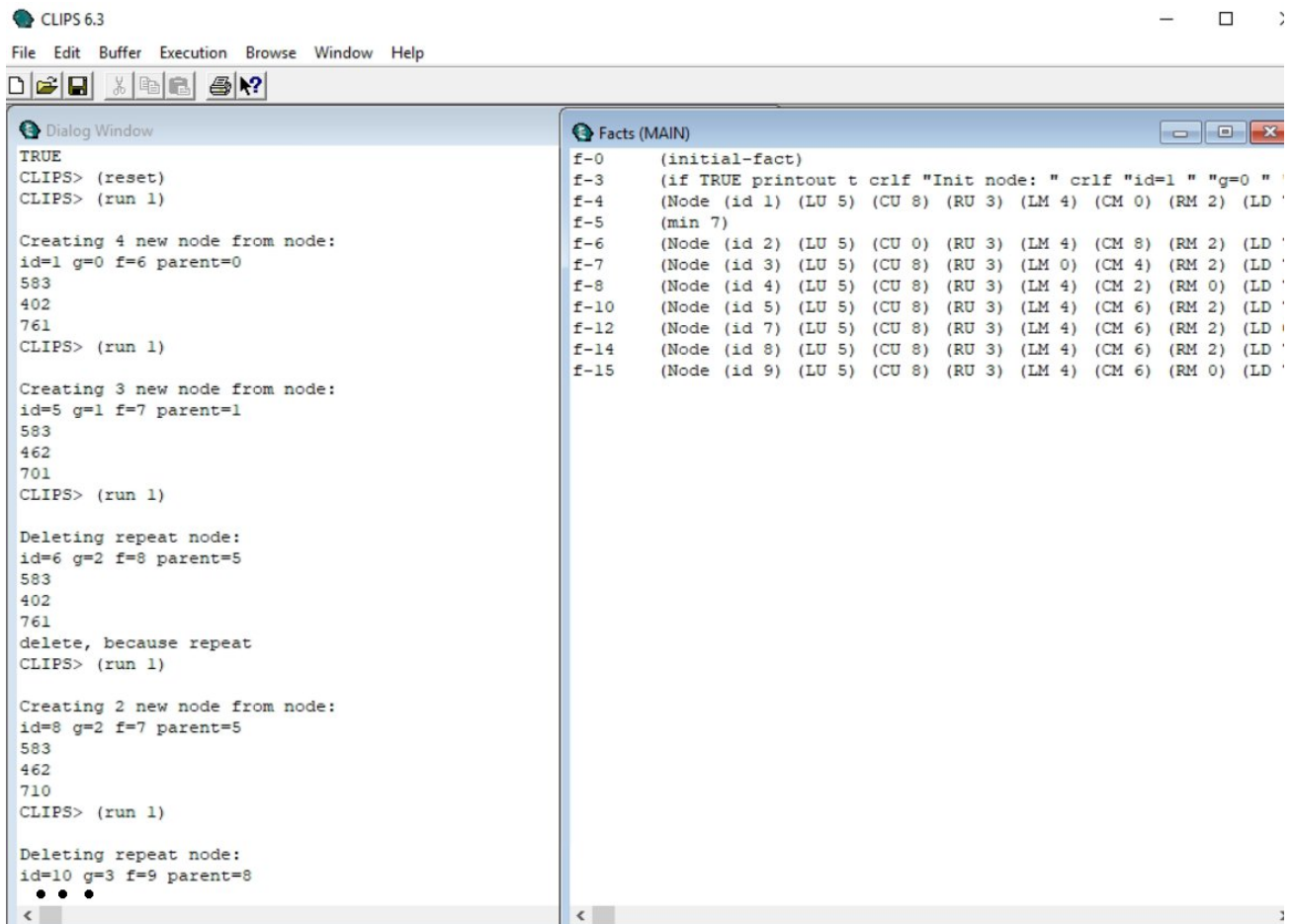


Рисунок 2 — Пример протокола выполнения программы в пошаговом режиме с эвристической функцией $h1$.

```

Defining defrule: make_new_from_RD =j+j+j
TRUE
CLIPS> (reset)
CLIPS> (run)

```

```

=====
id=4390 g=22 f=22 parent=4387
123
456
780
^
id=4387 g=21 f=23 parent=4385
123
456
708
^
id=4385 g=20 f=24 parent=4380
123
456
078
^
id=4380 g=19 f=25 parent=4377
123
056
478
^
id=4377 g=18 f=24 parent=4374
123
506
478
^
id=4374 g=17 f=25 parent=3555
103
526
478
...

```

```

f-0      (initial-fact)
f-3      (if FALSE printout t crlf "Init node: " crlf "id=1 " "g=0 "
f-7478   (min 23)
f-7483   (Node (id 4390) (LU 1) (CU 2) (RU 3) (LM 4) (CM 5) (RM 6) (LD
f-7484   (Node (id 4387) (LU 1) (CU 2) (RU 3) (LM 4) (CM 5) (RM 6) (LD
f-7485   (Node (id 4385) (LU 1) (CU 2) (RU 3) (LM 4) (CM 5) (RM 6) (LD
f-7486   (Node (id 4380) (LU 1) (CU 2) (RU 3) (LM 0) (CM 5) (RM 6) (LD
f-7487   (Node (id 4377) (LU 1) (CU 2) (RU 3) (LM 5) (CM 0) (RM 6) (LD
f-7488   (Node (id 4374) (LU 1) (CU 0) (RU 3) (LM 5) (CM 2) (RM 6) (LD
f-7489   (Node (id 3555) (LU 0) (CU 1) (RU 3) (LM 5) (CM 2) (RM 6) (LD
f-7490   (Node (id 1144) (LU 5) (CU 1) (RU 3) (LM 0) (CM 2) (RM 6) (LD
f-7491   (Node (id 236) (LU 5) (CU 1) (RU 3) (LM 4) (CM 2) (RM 6) (LD
f-7492   (Node (id 159) (LU 5) (CU 1) (RU 3) (LM 4) (CM 2) (RM 6) (LD
f-7493   (Node (id 157) (LU 5) (CU 1) (RU 3) (LM 4) (CM 2) (RM 6) (LD
f-7494   (Node (id 153) (LU 5) (CU 1) (RU 3) (LM 4) (CM 2) (RM 0) (LD
f-7495   (Node (id 149) (LU 5) (CU 1) (RU 3) (LM 4) (CM 0) (RM 2) (LD
f-7496   (Node (id 100) (LU 5) (CU 0) (RU 3) (LM 4) (CM 1) (RM 2) (LD
f-7497   (Node (id 43) (LU 5) (CU 3) (RU 0) (LM 4) (CM 1) (RM 2) (LD
f-7498   (Node (id 41) (LU 5) (CU 3) (RU 2) (LM 4) (CM 1) (RM 0) (LD
f-7499   (Node (id 34) (LU 5) (CU 3) (RU 2) (LM 4) (CM 0) (RM 1) (LD
f-7500   (Node (id 33) (LU 5) (CU 3) (RU 2) (LM 4) (CM 8) (RM 1) (LD
f-7501   (Node (id 31) (LU 5) (CU 3) (RU 2) (LM 4) (CM 8) (RM 1) (LD
f-7502   (Node (id 28) (LU 5) (CU 3) (RU 2) (LM 4) (CM 8) (RM 0) (LD
f-7503   (Node (id 21) (LU 5) (CU 3) (RU 0) (LM 4) (CM 8) (RM 2) (LD
f-7504   (Node (id 2) (LU 5) (CU 0) (RU 3) (LM 4) (CM 8) (RM 2) (LD 7
f-7505   (Node (id 1) (LU 5) (CU 8) (RU 3) (LM 4) (CM 0) (RM 2) (LD 7

```

Dialog Window

```

^
id=33 g=5 f=15 parent=31
532
481
706
^
id=31 g=4 f=14 parent=28
532
481
760
^
id=28 g=3 f=15 parent=21
532
480
761
^
id=21 g=2 f=16 parent=2
530
482
761
^
id=2 g=1 f=15 parent=1
503
482
761
^
id=1 g=0 f=14 parent=0
583
402
761

Solution finded!
Nodes number: 2641
Inter count: 1622
CLIPS>

```

```

t)
intout t crlf "Init node: " crlf
90) (LU 1) (CU 2) (RU 3) (LM 4)
87) (LU 1) (CU 2) (RU 3) (LM 4)
85) (LU 1) (CU 2) (RU 3) (LM 4)
80) (LU 1) (CU 2) (RU 3) (LM 0)
77) (LU 1) (CU 2) (RU 3) (LM 5)
74) (LU 1) (CU 0) (RU 3) (LM 5)
55) (LU 0) (CU 1) (RU 3) (LM 5)
44) (LU 5) (CU 1) (RU 3) (LM 0)
6) (LU 5) (CU 1) (RU 3) (LM 4)
9) (LU 5) (CU 1) (RU 3) (LM 4)
7) (LU 5) (CU 1) (RU 3) (LM 4)
3) (LU 5) (CU 1) (RU 3) (LM 4)
9) (LU 5) (CU 1) (RU 3) (LM 4)
0) (LU 5) (CU 0) (RU 3) (LM 4)
) (LU 5) (CU 3) (RU 0) (LM 4)
) (LU 5) (CU 3) (RU 2) (LM 4)
) (LU 5) (CU 3) (RU 2) (LM 4)
) (LU 5) (CU 3) (RU 2) (LM 4)
) (LU 5) (CU 3) (RU 2) (LM 4)
) (LU 5) (CU 3) (RU 0) (LM 4)
) (LU 5) (CU 0) (RU 3) (LM 4)
) (LU 5) (CU 8) (RU 3) (LM 4)

```

Рисунок 3 — Пример протокола выполнения программы в сквозном режиме с эвристической функцией h2.

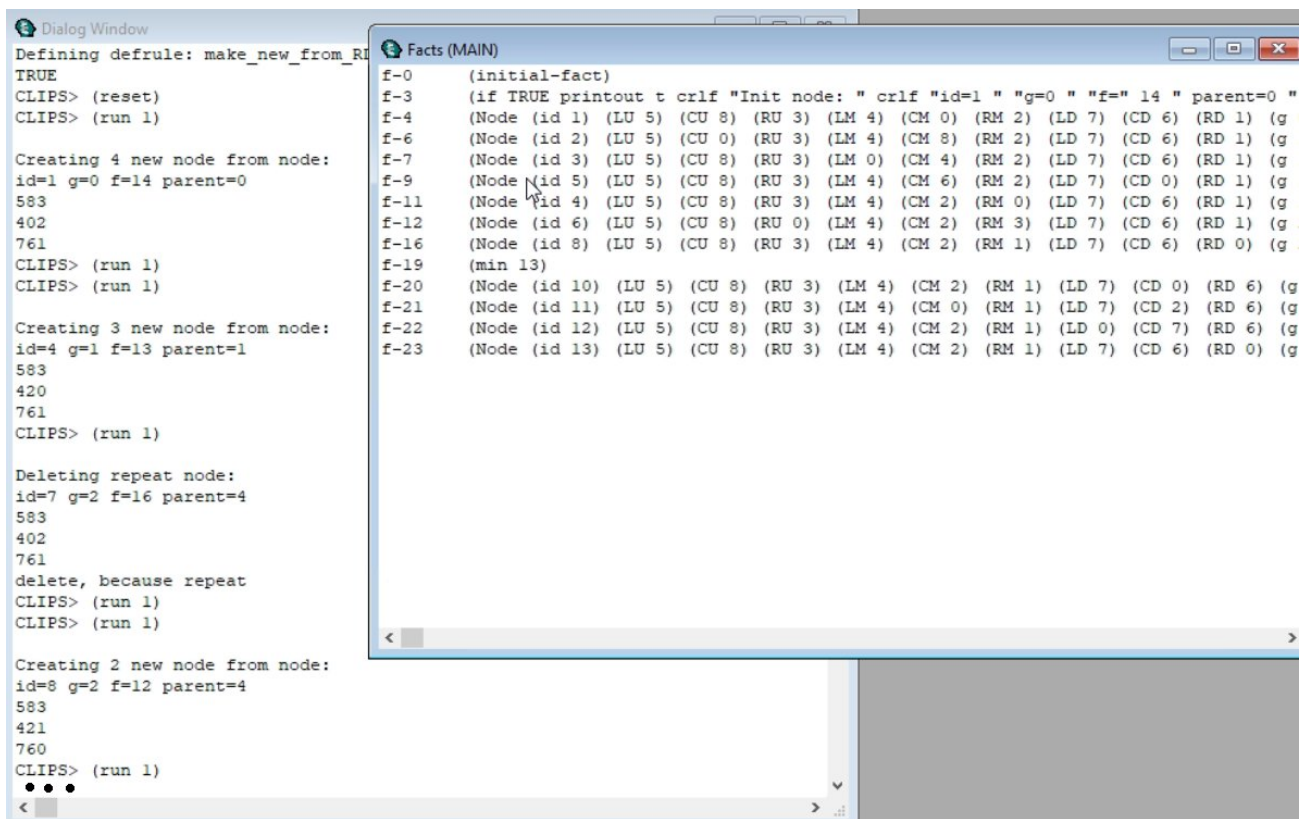


Рисунок 4 — Пример протокола выполнения программы в пошаговом режиме с эвристической функцией h_2 .

Экспериментальные оценки временной и ёмкостной сложности представлены в таблице 1.

Таблица 1 — Сравнительные оценки

	Временная	Ёмкостная
Python lab2, h1 (эксперимент)	1.4 s	-
Python lab2, h2 (эксперимент)	0.3 s	-
Python lab2, h1 (шаги, узлы)	10607 шагов	15966 узлов
Python lab2, h2 (шаги, узлы)	1841 шагов	2987 узлов
CLIPS lab4, h1 (эксперимент)	107 s	-
CLIPS lab4, h2 (эксперимент)	7 s	-
CLIPS lab4, h1 (шаги, узлы)	8017 шагов	12134 узлов
CLIPS lab4, h2 (шаги, узлы)	1622 шагов	2641 узлов

Из экспериментальных замеров видно, что при использовании эвристической функции h2, временная и ёмкостная сложность падает на порядок и в реализации задачи в лабораторной работе 2, и в реализации задачи в лабораторной работе 4, так как эвристическая функция h2 явно доминирует над h1 ($h2(x)$ всегда больше, чем $h1(x)$, или равна ей).

В итоге длины цепочек решения получились следующие:

- Эвристической функция h1: 22 узла
- Эвристической функция h2: 22 узла

Вывод

В ходе выполнения лабораторной работы было произведено практическое закрепление теоретических основ информированного (эвристического) поиска с использованием продукционного программирования в среде CLIPS.

Использованные источники

1. Рассел С, Норвиг П. Искусственный интеллект: современный подход, 2-е изд., М. «Вильямс», 2006.
2. Лекции “Логический вывод и стратегии разрешения конфликтов в системе CLIPS” и “Реализация поиска для головоломки восьмёрка в среде CLIPS” по предмету “Введение в искусственный интеллект”.

Приложение

Листинг:

(defglobal

?*HX* = 2 ; 1 if h1 or 2 if h2

?*DEBUG* = 0 ; 1 if пошаговый режим, 0 if сквозной режим

?*ID* = 0

?*NODE_COUNT* = 1 ; 1, потому что рутовый есть изначально

?*ITER_COUNT* = 0

?*init_LU* = 5

?*init_CU* = 8

?*init_RU* = 3

?*init_LM* = 4

?*init_CM* = 0

?*init_RM* = 2

?*init_LD* = 7

?*init_CD* = 6

?*init_RD* = 1

?*goal_LU* = 1

?*goal_CU* = 2

?*goal_RU* = 3

?*goal_LM* = 4

?*goal_CM* = 5

?*goal_RM* = 6

?*goal_LD* = 7

?*goal_CD* = 8

?*goal_RD* = 0

);

; Шаблон узла

(deftemplate Node

(slot id(type NUMBER) (default 0)) ; индификатор

(slot LU (type NUMBER)) ; left up

(slot CU (type NUMBER)) ; center up

(slot RU (type NUMBER)) ; right up

(slot LM (type NUMBER)) ; left mid

(slot CM (type NUMBER)) ; center mid

(slot RM (type NUMBER)) ; right mid

(slot LD (type NUMBER)) ; left down

(slot CD (type NUMBER)) ; center down

(slot RD (type NUMBER)) ; right down

(slot g (type NUMBER)) ; cost, depth

(slot status(type NUMBER) (default 0)) ; статус вершины: 0 – не раскрыта, 1 – раскрыта, 2 – соответствует решению

(slot parent (type NUMBER)) ; id родителя

(slot f (type NUMBER)) ; значение целевой функции для данной вершины $f = g + h$

);

(deffunction get_next_ID()

(bind ?*ID* (+ ?*ID* 1)) ;; инкрементируем ID

```
(return ?*ID*);
```

```
);
```

```
(deffunction h1( ?cur_LU ?cur_CU ?cur_RU
```

```
    ?cur_LM ?cur_CM ?cur_RM
```

```
    ?cur_LD ?cur_CD ?cur_RD)
```

```
(bind ?a 0);
```

```
(if (not (= ?cur_LU ?*goal_LU*)) then (bind ?a (+ ?a 1)))
```

```
(if (not (= ?cur_CU ?*goal_CU*)) then (bind ?a (+ ?a 1)))
```

```
(if (not (= ?cur_RU ?*goal_RU*)) then (bind ?a (+ ?a 1)))
```

```
(if (not (= ?cur_LM ?*goal_LM*)) then (bind ?a (+ ?a 1)))
```

```
(if (not (= ?cur_CM ?*goal_CM*)) then (bind ?a (+ ?a 1)))
```

```
(if (not (= ?cur_RM ?*goal_RM*)) then (bind ?a (+ ?a 1)))
```

```
(if (not (= ?cur_LD ?*goal_LD*)) then (bind ?a (+ ?a 1)))
```

```
(if (not (= ?cur_CD ?*goal_CD*)) then (bind ?a (+ ?a 1)))
```

```
(if (not (= ?cur_RD ?*goal_RD*)) then (bind ?a (+ ?a 1)))
```

```
(return ?a);
```

```
);
```

```
(deffunction manhattan(?v ?i ?j)
```

```
(if (= ?v ?*goal_LU*) then (bind ?i_g 0));
```

```
(if (= ?v ?*goal_LU*) then (bind ?j_g 0));
```

```
(if (= ?v ?*goal_CU*) then (bind ?i_g 0));
```

```
(if (= ?v ?*goal_CU*) then (bind ?j_g 1));
```

```
(if (= ?v ?*goal_RU*) then (bind ?i_g 0));
```

```
(if (= ?v ?*goal_RU*) then (bind ?j_g 2));
```

```

(if (= ?v ?*goal_LM*) then (bind ?i_g 1));
(if (= ?v ?*goal_LM*) then (bind ?j_g 0));
(if (= ?v ?*goal_CM*) then (bind ?i_g 1));
(if (= ?v ?*goal_CM*) then (bind ?j_g 1));
(if (= ?v ?*goal_RM*) then (bind ?i_g 1));
(if (= ?v ?*goal_RM*) then (bind ?j_g 2));
(if (= ?v ?*goal_LD*) then (bind ?i_g 2));
(if (= ?v ?*goal_LD*) then (bind ?j_g 0));
(if (= ?v ?*goal_CD*) then (bind ?i_g 2));
(if (= ?v ?*goal_CD*) then (bind ?j_g 1));
(if (= ?v ?*goal_RD*) then (bind ?i_g 2));
(if (= ?v ?*goal_RD*) then (bind ?j_g 2));
(return (+ (abs (- ?i ?i_g)) (abs (- ?j ?j_g))));
)

```

```

(deffunction h2(  ?cur_LU ?cur_CU ?cur_RU
                  ?cur_LM ?cur_CM ?cur_RM
                  ?cur_LD ?cur_CD ?cur_RD)
(bind ?a 0);
(bind ?a (+ ?a (manhattan ?cur_LU 0 0)));
(bind ?a (+ ?a (manhattan ?cur_CU 0 1)));
(bind ?a (+ ?a (manhattan ?cur_RU 0 2)));
(bind ?a (+ ?a (manhattan ?cur_LM 1 0)));
(bind ?a (+ ?a (manhattan ?cur_CM 1 1)));
(bind ?a (+ ?a (manhattan ?cur_RM 1 2)));
(bind ?a (+ ?a (manhattan ?cur_LD 2 0)));

```

```

(bind ?a (+ ?a (manhattan ?cur_CD 2 1)));

(bind ?a (+ ?a (manhattan ?cur_RD 2 2)));


(return ?a);

);


(deffunction calc_f(?g
    ?cur_LU ?cur_CU ?cur_RU
    ?cur_LM ?cur_CM ?cur_RM
    ?cur_LD ?cur_CD ?cur_RD)
(bind ?a ?g)
(if (= ?*HX* 1) then
    (bind ?a (+ ?a
(h1 ?cur_LU ?cur_CU ?cur_RU ?cur_LM ?cur_CM ?cur_RM ?cur_LD ?cur_CD ?cur_RD))))
)
(if (= ?*HX* 2) then
    (bind ?a (+ ?a
(h2 ?cur_LU ?cur_CU ?cur_RU ?cur_LM ?cur_CM ?cur_RM ?cur_LD ?cur_CD ?cur_RD))))
)
(return ?a);

);


(deffacts initial
(Node (id (get_next_ID))
    (LU ?*init_LU*) (CU ?*init_CU*) (RU ?*init_RU*)
    (LM ?*init_LM*) (CM ?*init_CM*) (RM ?*init_RM*)
    (LD ?*init_LD*) (CD ?*init_CD*) (RD ?*init_RD*)
    (g 0) (parent 0)

```

```

    (f (calc_f
0 ?*in
it_L
U* ?*init_CU* ?*i
nit_RU* ?*init_LM* ?*init_CM* ?*init_R
M* ?*init_LD* ?*init_CD* ?*init_RD*))

)

(min (calc_f
0 ?*in
it_L
U* ?*init_CU* ?*i
nit_RU* ?*init_LM* ?*init_CM* ?*init_R
M* ?*init_LD* ?*init_CD* ?*init_RD*))

(if (= ?*DEBUG* 1)

    printout t crlf "Init node: " crlf

    "id=1 " "g=0 " "f=" (calc_f
0 ?*in
it_L
U* ?*init_CU* ?*i
nit_RU* ?*init_LM* ?*init_CM* ?*init_R
M* ?*init_LD* ?*init_CD* ?*init_RD*) " parent=0 " crlf

    ?*init_LU* ?*init_CU* ?*init_RU* crlf ?*init_LM* ?*init_CM* ?*init_RM*
crlf ?*init_LD* ?*init_CD* ?*init_RD* crlf

)

);

;;; EKCnEPT EKCnEPTA BuDuT u3DAJIEKA,

;;; HOWA EKCnEPTA T9)l(EJlA,

;;; A CEPDcE CDEJlAHO u3 nECKA...

(defrule test_goal

(declare (salience 500))

?f_addr <- (Node (id ?v_id) (LU ?v_LU) (CU ?v_CU) (RU ?v_RU)

                (LM ?v_LM) (CM ?v_CM) (RM ?v_RM)

```

```

        (LD ?v_LD) (CD ?v_CD) (RD ?v_RD)

        (g ?v_g) (status ~2) (parent ?v_parent) (f ?v_f)

    )

(test (= ?v_LU ?*goal_LU*));
(test (= ?v_CU ?*goal_CU*));
(test (= ?v_RU ?*goal_RU*));
(test (= ?v_LM ?*goal_LM*));
(test (= ?v_CM ?*goal_CM*));
(test (= ?v_RM ?*goal_RM*));
(test (= ?v_LD ?*goal_LD*));
(test (= ?v_CD ?*goal_CD*));
(test (= ?v_RD ?*goal_RD*));

=>

(modify ?f_addr(status 2));

(printout t crlf "===== " crlf

"id=" ?v_id " g=" ?v_g " f=" ?v_f " parent=" ?v_parent crlf

    ?v_LU ?v_CU ?v_RU crlf

    ?v_LM ?v_CM ?v_RM crlf

    ?v_LD ?v_CD ?v_RD crlf);

);

(defrule stop_if_no_solution

(declare (salience 200))

(not (Node(status 0 | 2)))

=>

(halt);

```

```
(printout t crlf "No solution" crlf);  
);
```

```
(defrule stop_if_solution_finded  
(declare (salience 200))
```

```
(Node(status 2))
```

```
=>
```

```
(halt);  
(printout t crlf "Solution finded! " crlf);  
(printout t "Nodes number: " ?*NODE_COUNT* crlf);  
(printout t "Iter count: " ?*ITER_COUNT* crlf);  
);
```

```
(defrule fix_min  
(declare (salience 175))
```

```
?f_addr_min <- (min ?min)
```

```
(not (exists (Node (f ?F&:(= ?F ?min)) (status 0)) ))
```

```
=>
```

```
(retract ?f_addr_min);  
(assert (min (+ ?min 1)));  
);
```

```
(defrule find_min ; определение текущего минимума ЦФ  
(declare (salience 150))
```



```
?f_addr_min <- (min ?min)
```

```
(Node (f ?F&:(< ?F ?min)) (status 0)) ; Существование вершины, у которой  
; значение целевой функции меньше текущего min
```

```
=>
```

```
(retract ?f_addr_min) ;
```

```
(assert (min ?F)) ; обновить min
```

```
);
```

```
(defrule remove_repeats
```

```
(declare (salience 1000)) ; максимальный приоритет
```

```
?f_addr_1 <- (Node (id ?v_id_1) (LU ?v_LU_1) (CU ?v_CU_1) (RU ?v_RU_1)
```

```
(LM ?v_LM_1) (CM ?v_CM_1) (RM ?v_RM_1)
```

```
(LD ?v_LD_1) (CD ?v_CD_1) (RD ?v_RD_1)
```

```
(g ?v_g_1) (status 1) (parent ?v_parent_1) (f ?v_f_1)
```

```
)
```

```
?f_addr_2 <- (Node (id ?v_id_2&~?v_id_1)
```

```
(LU ?v_LU_2&:(= ?v_LU_1 ?v_LU_2)) (CU ?v_CU_2&:(= ?v_CU_1 ?v_CU_2))
```

```
(RU ?v_RU_2&:(= ?v_RU_1 ?v_RU_2))
```

```
(LM ?v_LM_2&:(= ?v_LM_1 ?v_LM_2)) (CM ?v_CM_2&:(= ?v_CM_1 ?v_CM_2))
```

```
(RM ?v_RM_2&:(= ?v_RM_1 ?v_RM_2))
```

```
(LD ?v_LD_2&:(= ?v_LD_1 ?v_LD_2)) (CD ?v_CD_2&:(= ?v_CD_1 ?v_CD_2))
```

```
(RD ?v_RD_2&:(= ?v_RD_1 ?v_RD_2))
```

```
(g ?v_g_2) (status 0) (parent ?v_parent_2) (f ?v_f_2)
```

```
)
```

```
=>
```

```

(if (= ?*DEBUG* 1) then
(printout t crlf "Deleting repeat node:" crlf);
(printout t "id=" ?v_id_2 " g=" ?v_g_2 " f=" ?v_f_2 " parent=" ?v_parent_2 crlf
      ?v_LU_2 ?v_CU_2 ?v_RU_2 crlf
      ?v_LM_2 ?v_CM_2 ?v_RM_2 crlf
      ?v_LD_2 ?v_CD_2 ?v_RD_2 crlf);
)
(if (<= ?v_f_1 ?v_f_2) then
      (retract ?f_addr_2) ; удаление повторной вершины с большей ЦФ
      (if (= ?*DEBUG* 1) then (printout t "delete, because repeat" crlf));
else
      (modify ?f_addr_1 (parent ?v_parent_2) (g ?v_g_2) (f ?v_f_2)) ; изменение с большей ЦФ
      (retract ?f_addr_2);
      (if (= ?*DEBUG* 1) then (printout t "delete and refresh prev repeat" crlf));
)
(bind ?*NODE_COUNT* (- ?*NODE_COUNT* 1));
)

(defrule show_answer
(declare (salience 500))

(Node (id ?v_id) (status 2) (parent ?v_pid))
?f_addr <- (Node      (LU ?v_LU) (CU ?v_CU) (RU ?v_RU)
              (LM ?v_LM) (CM ?v_CM) (RM ?v_RM)
              (LD ?v_LD) (CD ?v_CD) (RD ?v_RD)
              (id ?v_pid) (status ~2) (parent ?v_parent) (g ?v_g) (f ?v_f))

=>

```

```

(modify ?f_addr(status 2));
; (printout t ?v_id " <- " ?v_pid crlf);
(printout t "^" crlf);
(printout t "id=" ?v_pid " g=" ?v_g " f=" ?v_f " parent=" ?v_parent crlf
      ?v_LU ?v_CU ?v_RU crlf
      ?v_LM ?v_CM ?v_RM crlf
      ?v_LD ?v_CD ?v_RD crlf);
);

```

```

(defrule delete_not_answer

```

```

(declare (salience 400))

```

```

(Node(status 2))

```

```

?f_addr <- (Node(status ~2))

```

```

=>

```

```

(retract ?f_addr);

```

```

);

```

```

;;; C ETOpO MOMeHTA BECb MyP - ETO EKCnEPThbIE CuCTEMbl.

```

```

;;; Tbl EKCnEPT, 9 EKCnEPT, KOT EKCnEPT, CTOJI EKCnEPT...

```

```

(defrule make_new_from_LU

```

```

(declare (salience 100)) ; приоритет самый низкий!!

```

```

?f_addr_min <- (min ?min)

```

```

?f_addr_node <- (Node (id ?v_id) (LU 0) (CU ?v_CU) (RU ?v_RU)

```

```

      (LM ?v_LM) (CM ?v_CM) (RM ?v_RM)

```

```

      (LD ?v_LD) (CD ?v_CD) (RD ?v_RD)

```

```

      (g ?v_g) (status 0) (parent ?v_parent)

```

```

      (f ?v_f&:(= ?v_f ?min))

```

```

    )

=>

(if (= ?*DEBUG* 1) then (printout t crlf "Creating "
2 " new node from node: " crlf "id=" ?v_id " g=" ?v_g " f=" ?v_f " parent=" ?v_parent crlf

    (fact-slot-value ?f_addr_node LU) (fact-slot-value ?f_addr_node CU) (fact-slot-
value ?f_addr_node RU) crlf

    (fact-slot-value ?f_addr_node LM) (fact-slot-value ?f_addr_node CM) (fact-slot-
value ?f_addr_node RM) crlf

    (fact-slot-value ?f_addr_node LD) (fact-slot-value ?f_addr_node CD) (fact-slot-
value ?f_addr_node RD) crlf));

(modify ?f_addr_node(status 1));

(bind ?a1 (calc_f (+ ?v_g 1) ?v_CU 0 ?v_RU ?v_LM ?v_CM ?v_RM ?v_LD ?v_CD ?v_RD));
(assert (Node (id (get_next_ID)) (LU ?v_CU) (CU 0 ) (RU ?v_RU)

    (LM ?v_LM) (CM ?v_CM) (RM ?v_RM)

    (LD ?v_LD) (CD ?v_CD) (RD ?v_RD)

    (g (+ ?v_g 1)) (status 0) (parent ?v_id) (f ?a1)

    )

);

(bind ?a2 (calc_f (+ ?v_g 1) ?v_LM ?v_CU ?v_RU 0 ?v_CM ?v_RM ?v_LD ?v_CD ?v_RD));
(assert (Node (id (get_next_ID)) (LU ?v_LM) (CU ?v_CU) (RU ?v_RU)

    (LM 0) (CM ?v_CM) (RM ?v_RM)

    (LD ?v_LD) (CD ?v_CD) (RD ?v_RD)

    (g (+ ?v_g 1)) (status 0) (parent ?v_id) (f ?a2)

    )

);

```

```

(bind ?*NODE_COUNT* (+ ?*NODE_COUNT* 2));
(bind ?*ITER_COUNT* (+ ?*ITER_COUNT* 1));
);

```

```

(defrule make_new_from_CU
(declare (salience 100)) ; приоритет самый низкий!!
?f_addr_min <- (min ?min)

```

```

?f_addr_node <- (Node (id ?v_id) (LU ?v_LU) (CU 0) (RU ?v_RU)
                     (LM ?v_LM) (CM ?v_CM) (RM ?v_RM)
                     (LD ?v_LD) (CD ?v_CD) (RD ?v_RD)
                     (g ?v_g) (status 0) (parent ?v_parent)
                     (f ?v_f&:(= ?v_f ?min))
                    )

```

=>

```

(if (= ?*DEBUG* 1) then (printout t crlf "Creating "
3 " new node from node: " crlf "id=" ?v_id " g=" ?v_g " f=" ?v_f " parent=" ?v_parent crlf
    (fact-slot-value ?f_addr_node LU) (fact-slot-value ?f_addr_node CU) (fact-slot-
value ?f_addr_node RU) crlf
    (fact-slot-value ?f_addr_node LM) (fact-slot-value ?f_addr_node CM) (fact-slot-
value ?f_addr_node RM) crlf
    (fact-slot-value ?f_addr_node LD) (fact-slot-value ?f_addr_node CD) (fact-slot-
value ?f_addr_node RD) crlf));

(modify ?f_addr_node(status 1));

```

```

(bind ?a1 (calc_f (+ ?v_g 1) 0 ?v_LU ?v_RU ?v_LM ?v_CM ?v_RM ?v_LD ?v_CD ?v_RD));
(assert (Node (id (get_next_ID)) (LU 0) (CU ?v_LU) (RU ?v_RU)
              (LM ?v_LM) (CM ?v_CM) (RM ?v_RM)
              (LD ?v_LD) (CD ?v_CD) (RD ?v_RD)
              (g (+ ?v_g 1)) (status 0) (parent ?v_id) (f ?a1)
              )
);

```

```

(bind ?a2 (calc_f (+ ?v_g 1) ?v_LU ?v_CM ?v_RU ?v_LM 0 ?v_RM ?v_LD ?v_CD ?v_RD));
(assert (Node (id (get_next_ID)) (LU ?v_LU) (CU ?v_CM) (RU ?v_RU)
              (LM ?v_LM) (CM 0) (RM ?v_RM)
              (LD ?v_LD) (CD ?v_CD) (RD ?v_RD)
              (g (+ ?v_g 1)) (status 0) (parent ?v_id) (f ?a2)
              )
);

```

```

(bind ?a3 (calc_f (+ ?v_g 1) ?v_LU ?v_RU 0 ?v_LM ?v_CM ?v_RM ?v_LD ?v_CD ?v_RD));
(assert (Node (id (get_next_ID)) (LU ?v_LU) (CU ?v_RU) (RU 0)
              (LM ?v_LM) (CM ?v_CM) (RM ?v_RM)
              (LD ?v_LD) (CD ?v_CD) (RD ?v_RD)
              (g (+ ?v_g 1)) (status 0) (parent ?v_id) (f ?a3)
              )
);

```

```

(bind ?*NODE_COUNT* (+ ?*NODE_COUNT* 3));
(bind ?*ITER_COUNT* (+ ?*ITER_COUNT* 1));

```

```
);
```

```
(defrule make_new_from_RU
```

```
(declare (salience 100)) ; приоритет самый низкий!!
```

```
?f_addr_min <- (min ?min)
```

```
?f_addr_node <- (Node (id ?v_id) (LU ?v_LU) (CU ?v_CU) (RU 0)
```

```
      (LM ?v_LM) (CM ?v_CM) (RM ?v_RM)
```

```
      (LD ?v_LD) (CD ?v_CD) (RD ?v_RD)
```

```
      (g ?v_g) (status 0) (parent ?v_parent)
```

```
      (f ?v_f&:(= ?v_f ?min))
```

```
    )
```

```
=>
```

```
(if (= ?*DEBUG* 1) then (printout t crlf "Creating "
```

```
2 " new node from node: " crlf "id=" ?v_id " g=" ?v_g " f=" ?v_f " parent=" ?v_parent crlf
```

```
      (fact-slot-value ?f_addr_node LU) (fact-slot-value ?f_addr_node CU) (fact-slot-  
value ?f_addr_node RU) crlf
```

```
      (fact-slot-value ?f_addr_node LM) (fact-slot-value ?f_addr_node CM) (fact-slot-  
value ?f_addr_node RM) crlf
```

```
      (fact-slot-value ?f_addr_node LD) (fact-slot-value ?f_addr_node CD) (fact-slot-  
value ?f_addr_node RD) crlf));
```

```
(modify ?f_addr_node(status 1));
```

```
(bind ?a1 (calc_f (+ ?v_g 1) ?v_LU 0 ?v_CU ?v_LM ?v_CM ?v_RM ?v_LD ?v_CD ?v_RD));
```

```
(assert (Node (id (get_next_ID)) (LU ?v_LU) (CU 0) (RU ?v_CU)
```

```

(LM ?v_LM) (CM ?v_CM) (RM ?v_RM)

(LD ?v_LD) (CD ?v_CD) (RD ?v_RD)

(g (+ ?v_g 1)) (status 0) (parent ?v_id) (f ?a1)

)

);

(bind ?a2 (calc_f (+ ?v_g 1) ?v_LU ?v_CU ?v_RM ?v_LM ?v_CM 0 ?v_LD ?v_CD ?v_RD));

(assert (Node (id (get_next_ID)) (LU ?v_LU) (CU ?v_CU) (RU ?v_RM)

(LM ?v_LM) (CM ?v_CM) (RM 0)

(LD ?v_LD) (CD ?v_CD) (RD ?v_RD)

(g (+ ?v_g 1)) (status 0) (parent ?v_id) (f ?a2)

)

);

(bind ?*NODE_COUNT* (+ ?*NODE_COUNT* 2));

(bind ?*ITER_COUNT* (+ ?*ITER_COUNT* 1));

);

(defrule make_new_from_LM

(declare (salience 100)) ; приоритет самый низкий!!

?f_addr_min <- (min ?min)

?f_addr_node <- (Node (id ?v_id) (LU ?v_LU) (CU ?v_CU) (RU ?v_RU)

(LM 0) (CM ?v_CM) (RM ?v_RM)

(LD ?v_LD) (CD ?v_CD) (RD ?v_RD)

(g ?v_g) (status 0) (parent ?v_parent)

(f ?v_f&:(= ?v_f ?min))

```



```

    )

=>

(if (= ?*DEBUG* 1) then (printout t crlf "Creating "
3 " new node from node: " crlf "id=" ?v_id " g=" ?v_g " f=" ?v_f " parent=" ?v_parent crlf

    (fact-slot-value ?f_addr_node LU) (fact-slot-value ?f_addr_node CU) (fact-slot-
value ?f_addr_node RU) crlf

    (fact-slot-value ?f_addr_node LM) (fact-slot-value ?f_addr_node CM) (fact-slot-
value ?f_addr_node RM) crlf

    (fact-slot-value ?f_addr_node LD) (fact-slot-value ?f_addr_node CD) (fact-slot-
value ?f_addr_node RD) crlf));

(modify ?f_addr_node(status 1));

(bind ?a1 (calc_f (+ ?v_g 1) 0 ?v_CU ?v_RU ?v_LU ?v_CM ?v_RM ?v_LD ?v_CD ?v_RD));
(assert (Node (id (get_next_ID)) (LU 0) (CU ?v_CU) (RU ?v_RU)

    (LM ?v_LU) (CM ?v_CM) (RM ?v_RM)

    (LD ?v_LD) (CD ?v_CD) (RD ?v_RD)

    (g (+ ?v_g 1)) (status 0) (parent ?v_id) (f ?a1)

    )

);

(bind ?a2 (calc_f (+ ?v_g 1) ?v_LU ?v_CU ?v_RU ?v_CM 0 ?v_RM ?v_LD ?v_CD ?v_RD));
(assert (Node (id (get_next_ID)) (LU ?v_LU) (CU ?v_CU) (RU ?v_RU)

    (LM ?v_CM) (CM 0) (RM ?v_RM)

    (LD ?v_LD) (CD ?v_CD) (RD ?v_RD)

    (g (+ ?v_g 1)) (status 0) (parent ?v_id) (f ?a2)

    )

);

```

```

(bind ?a3 (calc_f (+ ?v_g 1) ?v_LU ?v_CU ?v_RU ?v_LD ?v_CM ?v_RM 0 ?v_CD ?v_RD));
(assert (Node (id (get_next_ID)) (LU ?v_LU) (CU ?v_CU) (RU ?v_RU)
              (LM ?v_LD) (CM ?v_CM) (RM ?v_RM)
              (LD 0) (CD ?v_CD) (RD ?v_RD)
              (g (+ ?v_g 1)) (status 0) (parent ?v_id) (f ?a3)
              )
);

```

```

(bind ?*NODE_COUNT* (+ ?*NODE_COUNT* 3));
(bind ?*ITER_COUNT* (+ ?*ITER_COUNT* 1));
);

```

```

(defrule make_new_from_CM
(declare (salience 100)) ; приоритет самый низкий!!
?f_addr_min <- (min ?min)

```

```

?f_addr_node <- (Node (id ?v_id) (LU ?v_LU) (CU ?v_CU) (RU ?v_RU)
                      (LM ?v_LM) (CM 0) (RM ?v_RM)
                      (LD ?v_LD) (CD ?v_CD) (RD ?v_RD)
                      (g ?v_g) (status 0) (parent ?v_parent)
                      (f ?v_f&:(= ?v_f ?min))
                      )

```

=>

```

(if (= ?*DEBUG* 1) then (printout t crlf "Creating "

```

```

4 " new node from node: " crlf "id=" ?v_id " g=" ?v_g " f=" ?v_f " parent=" ?v_parent crlf

    (fact-slot-value ?f_addr_node LU) (fact-slot-value ?f_addr_node CU) (fact-slot-
value ?f_addr_node RU) crlf

    (fact-slot-value ?f_addr_node LM) (fact-slot-value ?f_addr_node CM) (fact-slot-
value ?f_addr_node RM) crlf

    (fact-slot-value ?f_addr_node LD) (fact-slot-value ?f_addr_node CD) (fact-slot-
value ?f_addr_node RD) crlf));

(modify ?f_addr_node(status 1));

(bind ?a1 (calc_f (+ ?v_g 1) ?v_LU 0 ?v_RU ?v_LM ?v_CU ?v_RM ?v_LD ?v_CD ?v_RD));
(retract ?f_addr_min);
(assert (min ?a1));
(assert (Node (id (get_next_ID)) (LU ?v_LU) (CU 0) (RU ?v_RU)
    (LM ?v_LM) (CM ?v_CU) (RM ?v_RM)
    (LD ?v_LD) (CD ?v_CD) (RD ?v_RD)
    (g (+ ?v_g 1)) (status 0) (parent ?v_id) (f ?a1)
    )
);

(bind ?a2 (calc_f (+ ?v_g 1) ?v_LU ?v_CU ?v_RU 0 ?v_LM ?v_RM ?v_LD ?v_CD ?v_RD));
(assert (Node (id (get_next_ID)) (LU ?v_LU) (CU ?v_CU) (RU ?v_RU)
    (LM 0) (CM ?v_LM) (RM ?v_RM)
    (LD ?v_LD) (CD ?v_CD) (RD ?v_RD)
    (g (+ ?v_g 1)) (status 0) (parent ?v_id) (f ?a2)
    )
);

```

```

(bind ?a3 (calc_f (+ ?v_g 1) ?v_LU ?v_CU ?v_RU ?v_LM ?v_RM 0 ?v_LD ?v_CD ?v_RD));
(assert (Node (id (get_next_ID)) (LU ?v_LU) (CU ?v_CU) (RU ?v_RU)
              (LM ?v_LM) (CM ?v_RM) (RM 0)
              (LD ?v_LD) (CD ?v_CD) (RD ?v_RD)
              (g (+ ?v_g 1)) (status 0) (parent ?v_id) (f ?a3)
              )
);

```

```

(bind ?a4 (calc_f (+ ?v_g 1) ?v_LU ?v_CU ?v_RU ?v_LM ?v_CD ?v_RM ?v_LD 0 ?v_RD));
(assert (Node (id (get_next_ID)) (LU ?v_LU) (CU ?v_CU) (RU ?v_RU)
              (LM ?v_LM) (CM ?v_CD) (RM ?v_RM)
              (LD ?v_LD) (CD 0) (RD ?v_RD)
              (g (+ ?v_g 1)) (status 0) (parent ?v_id) (f ?a4)
              )
);

```

```

(bind ?*NODE_COUNT* (+ ?*NODE_COUNT* 4));
(bind ?*ITER_COUNT* (+ ?*ITER_COUNT* 1));
);

```

```

(defrule make_new_from_RM
(declare (salience 100)) ; приоритет самый низкий!!
?f_addr_min <- (min ?min)

?f_addr_node <- (Node (id ?v_id) (LU ?v_LU) (CU ?v_CU) (RU ?v_RU)

```

```

(LM ?v_LM) (CM ?v_CM) (RM 0)

(LD ?v_LD) (CD ?v_CD) (RD ?v_RD)

(g ?v_g) (status 0) (parent ?v_parent)

(f ?v_f&:(= ?v_f ?min))

)

=>

(if (= ?*DEBUG* 1) then (printout t crlf "Creating "

3 " new node from node: " crlf "id=" ?v_id " g=" ?v_g " f=" ?v_f " parent=" ?v_parent crlf

(fact-slot-value ?f_addr_node LU) (fact-slot-value ?f_addr_node CU) (fact-slot-
value ?f_addr_node RU) crlf

(fact-slot-value ?f_addr_node LM) (fact-slot-value ?f_addr_node CM) (fact-slot-
value ?f_addr_node RM) crlf

(fact-slot-value ?f_addr_node LD) (fact-slot-value ?f_addr_node CD) (fact-slot-
value ?f_addr_node RD) crlf));

(modify ?f_addr_node(status 1));

(bind ?a1 (calc_f (+ ?v_g 1) ?v_LU ?v_CU 0 ?v_LM ?v_CM ?v_RU ?v_LD ?v_CD ?v_RD));
(assert (Node (id (get_next_ID)) (LU ?v_LU) (CU ?v_CU) (RU 0)

(LM ?v_LM) (CM ?v_CM) (RM ?v_RU)

(LD ?v_LD) (CD ?v_CD) (RD ?v_RD)

(g (+ ?v_g 1)) (status 0) (parent ?v_id) (f ?a1)

)

);

(bind ?a2 (calc_f (+ ?v_g 1) ?v_LU ?v_CU ?v_RU ?v_LM 0 ?v_CM ?v_LD ?v_CD ?v_RD));
(assert (Node (id (get_next_ID)) (LU ?v_LU) (CU ?v_CU) (RU ?v_RU)

(LM ?v_LM) (CM 0) (RM ?v_CM)

```

```

        (LD ?v_LD) (CD ?v_CD) (RD ?v_RD)
      (g (+ ?v_g 1)) (status 0) (parent ?v_id) (f ?a2)
    )
  );

(bind ?a3 (calc_f (+ ?v_g 1) ?v_LU ?v_CU ?v_RU ?v_LM ?v_CM ?v_RD ?v_LD ?v_CD 0));
(assert (Node (id (get_next_ID)) (LU ?v_LU) (CU ?v_CU) (RU ?v_RU)
              (LM ?v_LM) (CM ?v_CM) (RM ?v_RM)
              (LD ?v_LD) (CD ?v_CD) (RD 0)
              (g (+ ?v_g 1)) (status 0) (parent ?v_id) (f ?a3)
              )
);

(bind ?*NODE_COUNT* (+ ?*NODE_COUNT* 3));
(bind ?*ITER_COUNT* (+ ?*ITER_COUNT* 1));
);

;;; |---| 3) |---|
(defrule make_new_from_LD
(declare (salience 100)) ; приоритет самый низкий!!
?f_addr_min <- (min ?min)

?f_addr_node <- (Node (id ?v_id) (LU ?v_LU) (CU ?v_CU) (RU ?v_RU)
                      (LM ?v_LM) (CM ?v_CM) (RM ?v_RM)
                      (LD 0) (CD ?v_CD) (RD ?v_RD)
                      (g ?v_g) (status 0) (parent ?v_parent)
                      (f ?v_f&:(= ?v_f ?min))
                      )

```

=>

```
(if (= ?*DEBUG* 1) then (printout t crlf "Creating "
2 " new node from node: " crlf "id=" ?v_id " g=" ?v_g " f=" ?v_f " parent=" ?v_parent crlf
    (fact-slot-value ?f_addr_node LU) (fact-slot-value ?f_addr_node CU) (fact-slot-
value ?f_addr_node RU) crlf
    (fact-slot-value ?f_addr_node LM) (fact-slot-value ?f_addr_node CM) (fact-slot-
value ?f_addr_node RM) crlf
    (fact-slot-value ?f_addr_node LD) (fact-slot-value ?f_addr_node CD) (fact-slot-
value ?f_addr_node RD) crlf));

(modify ?f_addr_node(status 1));

(bind ?a1 (calc_f (+ ?v_g 1) ?v_LU ?v_CU ?v_RU 0 ?v_CM ?v_RM ?v_LM ?v_CD ?v_RD));
(assert (Node (id (get_next_ID)) (LU ?v_LU) (CU ?v_CU) (RU ?v_RU)
    (LM 0) (CM ?v_CM) (RM ?v_RM)
    (LD ?v_LM) (CD ?v_CD) (RD ?v_RD)
    (g (+ ?v_g 1)) (status 0) (parent ?v_id) (f ?a1)
    )
);

(bind ?a2 (calc_f (+ ?v_g 1) ?v_LU ?v_CU ?v_RU ?v_LM ?v_CM ?v_RM ?v_CD 0 ?v_RD));
(assert (Node (id (get_next_ID)) (LU ?v_LU) (CU ?v_CU) (RU ?v_RU)
    (LM ?v_LM) (CM ?v_CM) (RM ?v_RM)
    (LD ?v_CD) (CD 0) (RD ?v_RD)
    (g (+ ?v_g 1)) (status 0) (parent ?v_id) (f ?a2)
    )
);
```

```

(bind ?*NODE_COUNT* (+ ?*NODE_COUNT* 2));
(bind ?*ITER_COUNT* (+ ?*ITER_COUNT* 1));
);

```

```

(defrule make_new_from_CD
(declare (salience 100)) ; приоритет самый низкий!!
?f_addr_min <- (min ?min)

```

```

?f_addr_node <- (Node (id ?v_id) (LU ?v_LU) (CU ?v_CU) (RU ?v_RU)
                     (LM ?v_LM) (CM ?v_CM) (RM ?v_RM)
                     (LD ?v_LD) (CD 0) (RD ?v_RD)
                     (g ?v_g) (status 0) (parent ?v_parent)
                     (f ?v_f&:(= ?v_f ?min))
                     )

```

=>

```

(if (= ?*DEBUG* 1) then (printout t crlf "Creating "
3 " new node from node: " crlf "id=" ?v_id " g=" ?v_g " f=" ?v_f " parent=" ?v_parent crlf
    (fact-slot-value ?f_addr_node LU) (fact-slot-value ?f_addr_node CU) (fact-slot-
value ?f_addr_node RU) crlf
    (fact-slot-value ?f_addr_node LM) (fact-slot-value ?f_addr_node CM) (fact-slot-
value ?f_addr_node RM) crlf
    (fact-slot-value ?f_addr_node LD) (fact-slot-value ?f_addr_node CD) (fact-slot-
value ?f_addr_node RD) crlf));

(modify ?f_addr_node(status 1));

```



```

(bind ?a1 (calc_f (+ ?v_g 1) ?v_LU ?v_CU ?v_RU ?v_LM 0 ?v_RM ?v_LD ?v_CM ?v_RD));
(assert (Node (id (get_next_ID)) (LU ?v_LU) (CU ?v_CU) (RU ?v_RU)
              (LM ?v_LM) (CM 0) (RM ?v_RM)
              (LD ?v_LD) (CD ?v_CM) (RD ?v_RD)
              (g (+ ?v_g 1)) (status 0) (parent ?v_id) (f ?a1)
              )
);

```

```

(bind ?a2 (calc_f (+ ?v_g 1) ?v_LU ?v_CU ?v_RU ?v_LM ?v_CM ?v_RM 0 ?v_LD ?v_RD));
(assert (Node (id (get_next_ID)) (LU ?v_LU) (CU ?v_CU) (RU ?v_RU)
              (LM ?v_LM) (CM ?v_CM) (RM ?v_RM)
              (LD 0) (CD ?v_LD) (RD ?v_RD)
              (g (+ ?v_g 1)) (status 0) (parent ?v_id) (f ?a2)
              )
);

```

```

(bind ?a3 (calc_f (+ ?v_g 1) ?v_LU ?v_CU ?v_RU ?v_LM ?v_CM ?v_RM ?v_LD ?v_RD 0));
(assert (Node (id (get_next_ID)) (LU ?v_LU) (CU ?v_CU) (RU ?v_RU)
              (LM ?v_LM) (CM ?v_CM) (RM ?v_RM)
              (LD ?v_LD) (CD ?v_RD) (RD 0)
              (g (+ ?v_g 1)) (status 0) (parent ?v_id) (f ?a3)
              )
);

```

```

(bind ?*NODE_COUNT* (+ ?*NODE_COUNT* 3));
(bind ?*ITER_COUNT* (+ ?*ITER_COUNT* 1));
);

```

```

(defrule make_new_from_RD
(declare (salience 100)) ; приоритет самый низкий!!
?f_addr_min <- (min ?min)

?f_addr_node <- (Node (id ?v_id) (LU ?v_LU) (CU ?v_CU) (RU ?v_RU)
                     (LM ?v_LM) (CM ?v_CM) (RM ?v_RM)
                     (LD ?v_LD) (CD ?v_CD) (RD 0)
                     (g ?v_g) (status 0) (parent ?v_parent)
                     (f ?v_f&:(= ?v_f ?min))
                     )
=>
(if (= ?*DEBUG* 1) then (printout t crlf "Creating "
2 " new node from node: " crlf "id=" ?v_id " g=" ?v_g " f=" ?v_f " parent=" ?v_parent crlf
    (fact-slot-value ?f_addr_node LU) (fact-slot-value ?f_addr_node CU) (fact-slot-
value ?f_addr_node RU) crlf
    (fact-slot-value ?f_addr_node LM) (fact-slot-value ?f_addr_node CM) (fact-slot-
value ?f_addr_node RM) crlf
    (fact-slot-value ?f_addr_node LD) (fact-slot-value ?f_addr_node CD) (fact-slot-
value ?f_addr_node RD) crlf));

(modify ?f_addr_node(status 1));

(bind ?a1 (calc_f (+ ?v_g 1) ?v_LU ?v_CU ?v_RU ?v_LM ?v_CM 0 ?v_LD ?v_CD ?v_RM));
(assert (Node (id (get_next_ID)) (LU ?v_LU) (CU ?v_CU) (RU ?v_RU)
              (LM ?v_LM) (CM ?v_CM) (RM 0)

```

```

        (LD ?v_LD) (CD ?v_CD) (RD ?v_RM)
      (g (+ ?v_g 1)) (status 0) (parent ?v_id) (f ?a1)
    )
  );

(bind ?a2 (calc_f (+ ?v_g 1) ?v_LU ?v_CU ?v_RU ?v_LM ?v_CM ?v_RM ?v_LD 0 ?v_CD));
(assert (Node (id (get_next_ID)) (LU ?v_LU) (CU ?v_CU) (RU ?v_RU)
              (LM ?v_LM) (CM ?v_CM) (RM ?v_RM)
              (LD ?v_LD) (CD 0) (RD ?v_CD)
              (g (+ ?v_g 1)) (status 0) (parent ?v_id) (f ?a2)
            )
  );

(bind ?*NODE_COUNT* (+ ?*NODE_COUNT* 2));
(bind ?*ITER_COUNT* (+ ?*ITER_COUNT* 1));
);

```