**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра Вычислительной техники**

отчЁт

**по лабораторной работе №4**

**по дисциплине «Введение в искусственный интеллект»**

**Тема: «Методы информированного (эвристического) поиска в среде продукционного программирования CLIPS»**

|  |  |  |
| --- | --- | --- |
| Студенты гр. 9308 |  | Степовик В.С.  Соболев М.С.  Аюпов Р. Н. |
| Преподаватель |  | Родионов С.В. |

Санкт-Петербург

2022г.

**Цель работы**

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

**Постановка задачи**

1. Рассмотреть принципы эвристического поиска в среде продукционного программирования CLIPS на конкретном примере – головоломка 8-ка. Рассматриваем два вида эвристических функций h1 и h2 (Рассел С, Норвиг П. Искусственный интеллект: современный подход, 2-е изд., М. «Вильямс», 2006, стр. 167- 174):

– h1 – число фишек, стоящих не на своем месте;

– h2 – 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.

Описанные выше состояния изображены на рисунке 1 и 2 соответственно.

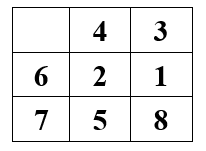


Рис. 1. Начальное состояние

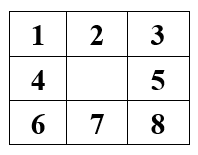


Рис. 2. Конечное состояние

**Распределение обязанностей**

// ПЕРЕПИСАТЬ РОЛИ ДЛЯ ЛАБОРАТОРНОЙ РАБОТЫ 4

Аюпов Ренат – модификация алгоритма слепого поиска, реализация алгоритма поиска A\*, реализация эвристической функции h1 (поиск фишек не на своих местах с присваиванием приоритета), обновление и добавление зависимостей структур данных.

Соболев Матвей – модификация алгоритма слепого поиска, реализация эвристической функции h2 (манхэттенское расстояние), обновление и добавление структур данных для задания лабораторной работы 2.

Степовик Виктор – сравнительная оценка временной сложности, описание классов, тестирование и отладка программы, доработка архитектуры программы для задания лабораторной работы 2.

**Описание выбранных структур данных, представление функции определения последователей**

В программной реализации для описания состояния будем использовать специально созданный шаблон «Node». Его слоты приведены ниже (рисунок 3).

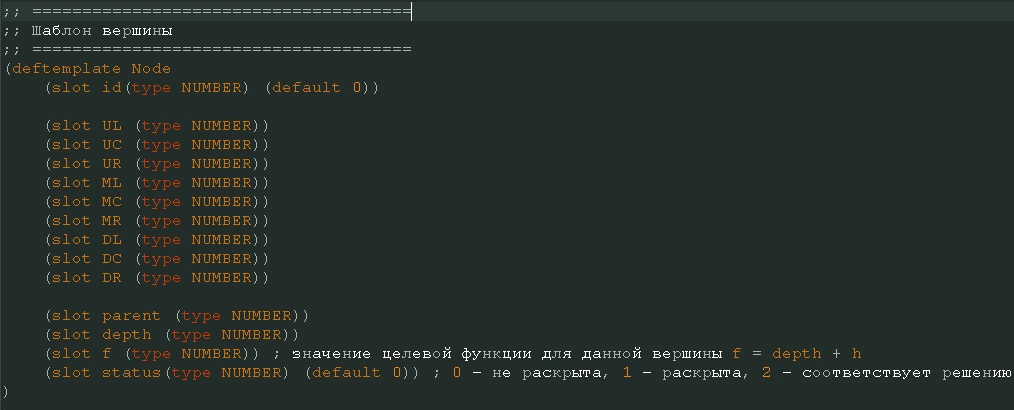


Рис. 3. Шаблон «Node»

Первым слотом представлен уникальный идентификатор вершины. Затем идут 9 слотов, которые описывают состояние (значения ячеек 3x3).

Далее идут следующие слоты:

1. «parent» – слот идентификатора родителя;
2. «depth» – глубина расположения вершины на дереве (корень 0);
3. «f» – значение оценочной функции для данного узла;
4. «status» – статус вершины (0 – не раскрыта, 1 – раскрыта, 2 – решение).

Создание начального и целевого состояний представлены на рисунке 4 и определены в начальной инициализации входных данных (глобально).

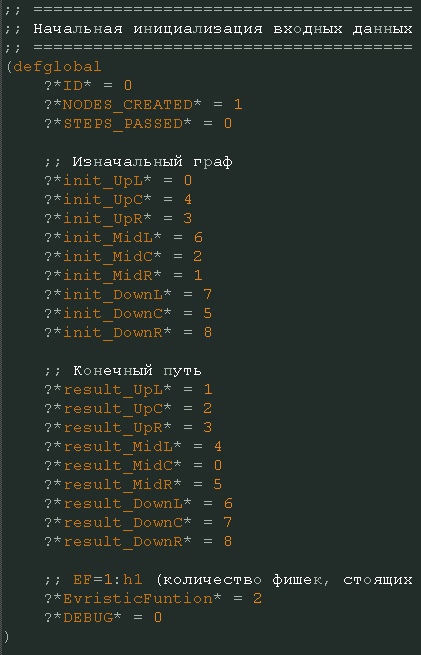


Рис. 4. Создание начального и целевого узлов

Далее для каждой из 9-ти ячеек (ситуаций) создадим своё уникальное правило, которое будет производить необходимые действия по порождению вершин потомков и вычисления оценочной функции для них. Также изменение статуса вершины родителя, обновления минимума оценочной функции среди всех раскрытых вершин. Пример реализации правила порождения вершин потомков представлен ниже.

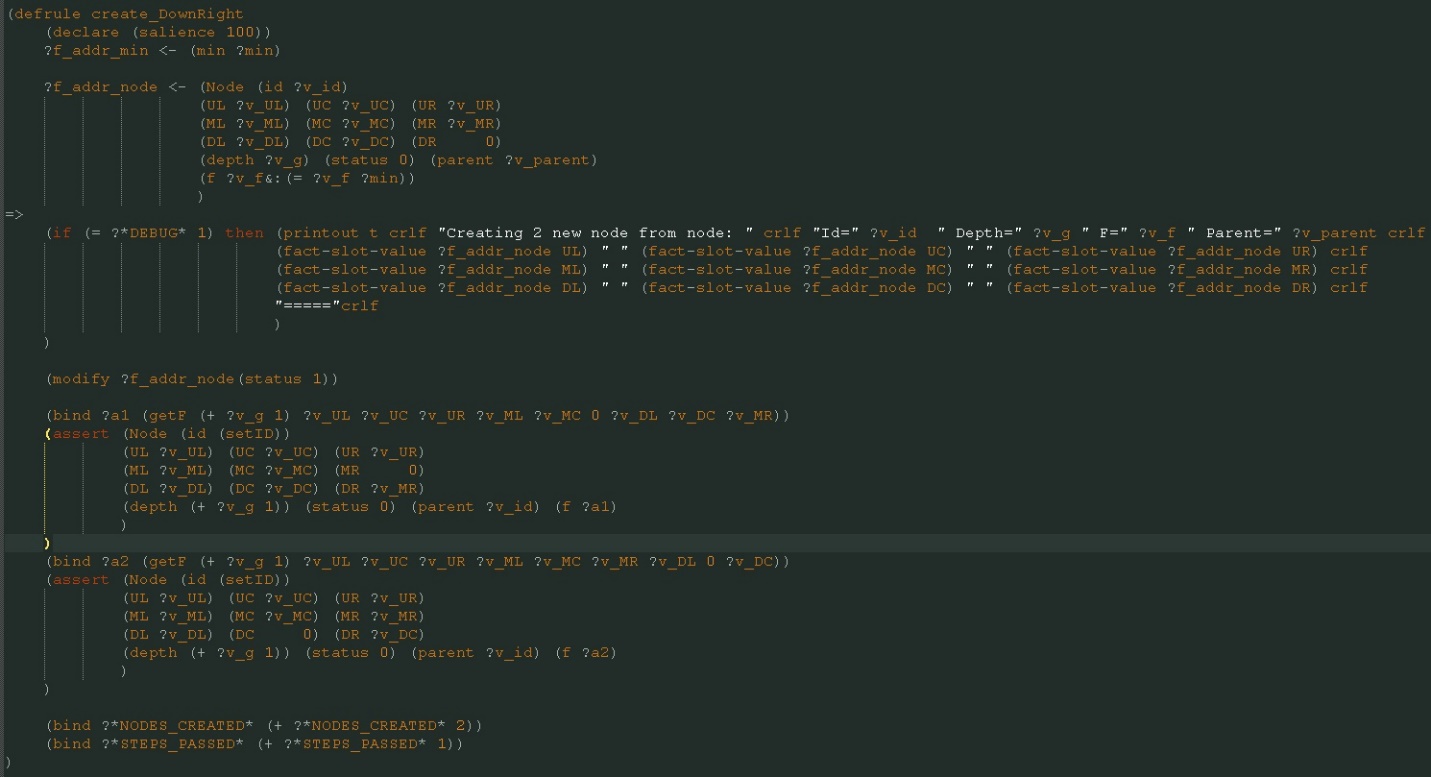


Рис. 5. Правила порождения вершин потомков для ячейки «DR» (правый нижний угол)

Для определения достижения конечного результата (конечного состояния) мы определили правило «result»

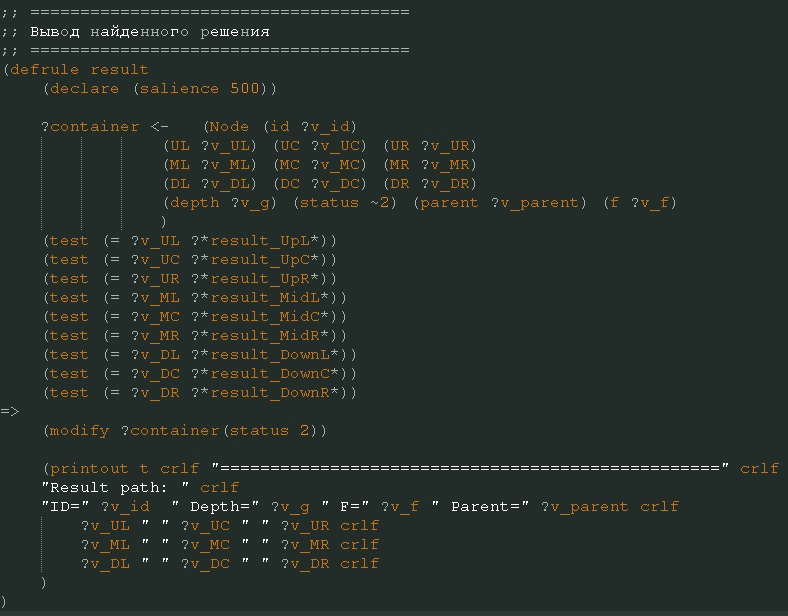


Рис. 6. Правила определения достижения конечного состояния

Для определения текущего минимума используется следующее правило. Стоит отметить, что данное правило имеет больший приоритет, чем правила создания дочерних узлов. Это решает важную проблему: с отслеживанием текущего минимума при его появлении, то есть мы сразу его обновляем.

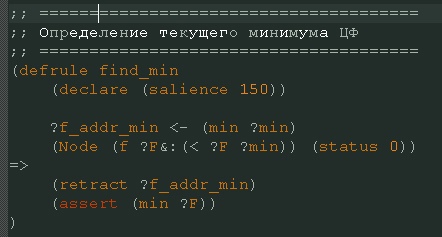


Рис. 7. Правило определения текущего минимума

Правило удаления для повторяющихся вершин представлено ниже. Оно обладает самым высоким приоритетом среди всех правил.

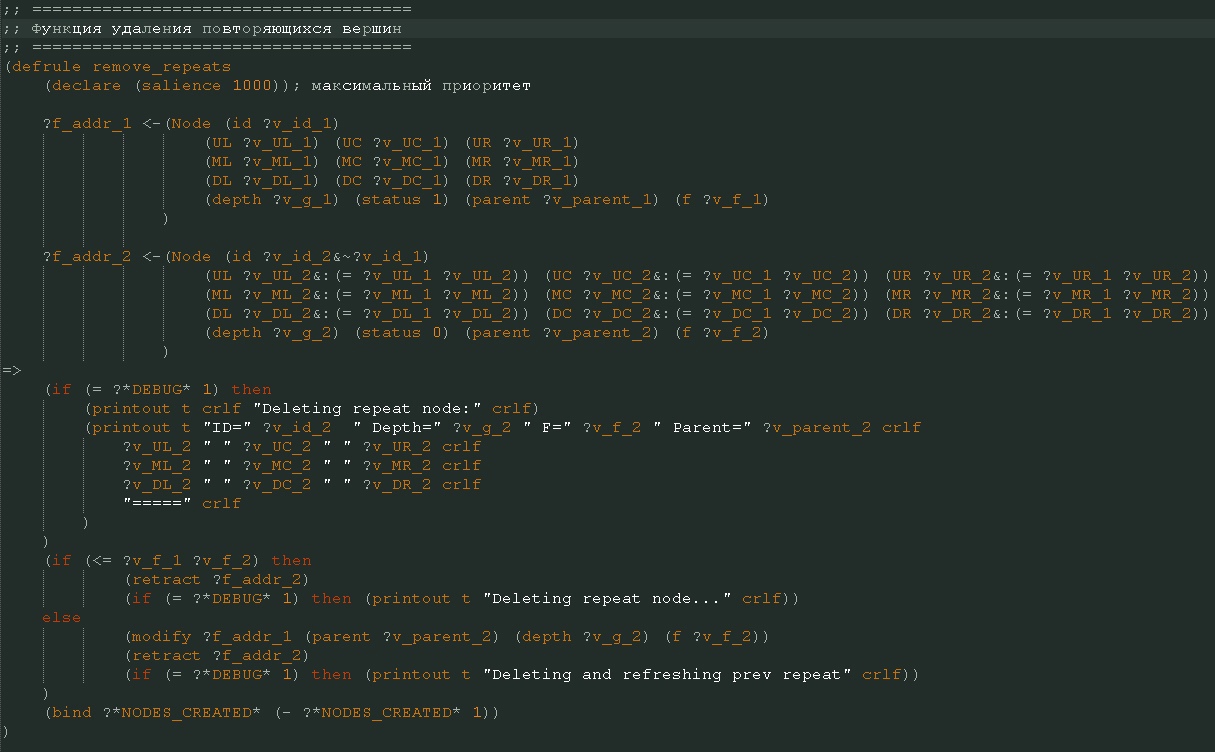


Рис. 8. Правило удаления для повторяющихся вершин

Затем приведём оценочные функции.

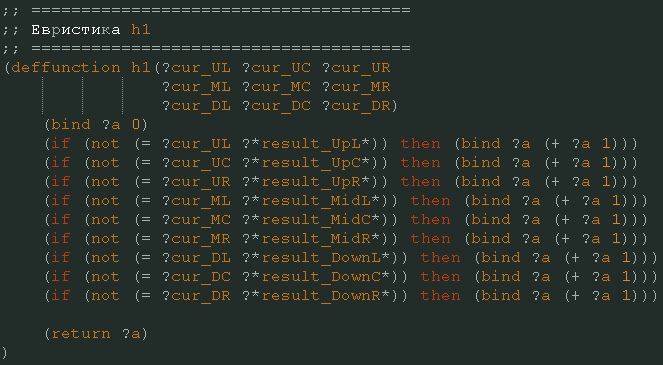


Рис. 9. Оценочная функция, использующая «h1»

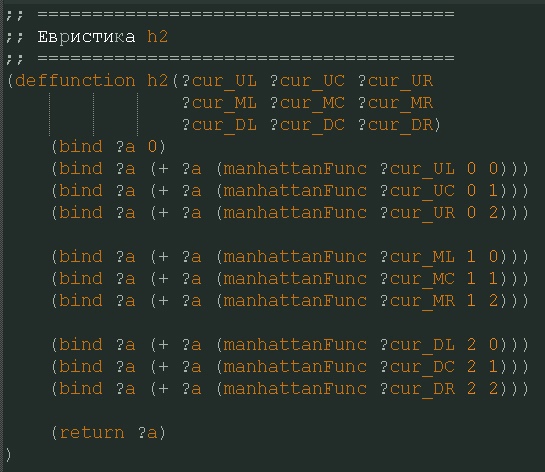


Рис. 10. Оценочная функция, использующая «h2»

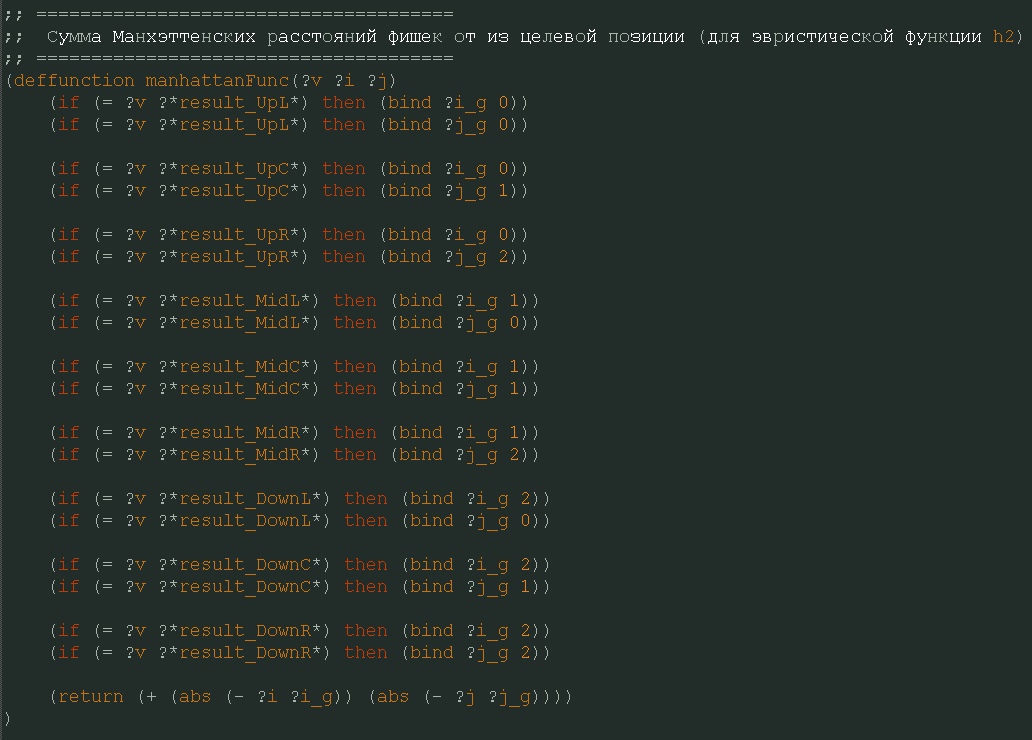


Рис. 11. Сумма Манхэттенских расстояний для «h2»

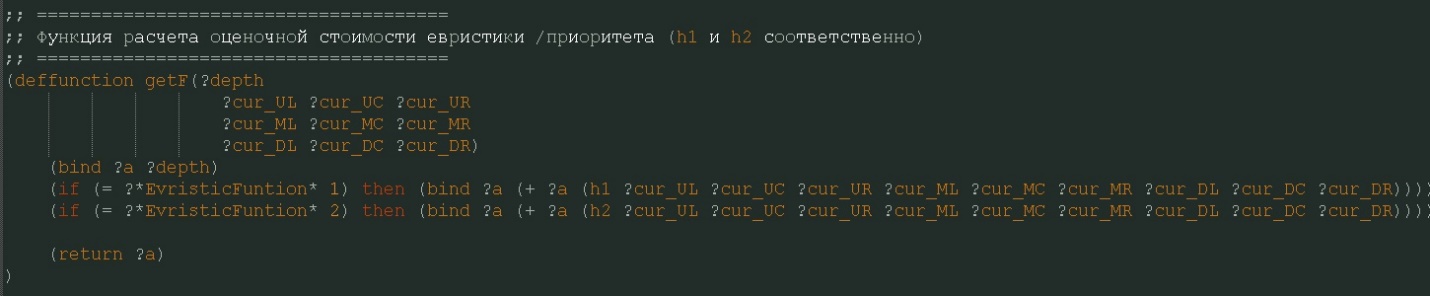


Рис. 12. Функция расчёта оценочной стоимости

**Описание алгоритма**

**Реализации в среде CLIPS**

1. Инициализация начальной вершины.
2. Исполнение правила создания дочерних вершин: порождение потомков из текущего состояния, определение их оценочных функций, установление текущей вершине статус 1.
3. Если существует узел с повторным состоянием, то удаляем его (если его целевая функция больше). В противном случае меняем родительскую вершину, стоимость и значение целевой функции на изначально раскрытый узел.
4. Если есть узел, состояние которого совпадает с целевым, то меняем ему статус на 2 (входит в решение задачи).
5. Если у раскрытого узла в данный момент имеется статус 2, то и его родителю тоже присваиваем статус 2.
6. Если у раскрытого узла в данный момент имеется статус 2, то остальные узлы с иным статусом удаляем.
7. Если у раскрытого узла в данный момент имеется статус 2, то исполняем правило «stop\_if\_solution\_finded», то есть решение найдено.
8. Если ни одна вершина не была раскрыта или не найдено вершин, соответствующих решению, то исполняем правило «stop\_if\_no\_solution», то есть принудительно завершаем работу программы.
9. Определение текущего минимума целевой функции.
10. Выполняем раскрытие рядом стоящих вершин относительно текущей (родительской).

**Пример работы программы**

Приведём пример работы программы эвристической функции «h1» (список фактов и диалоговое окно).

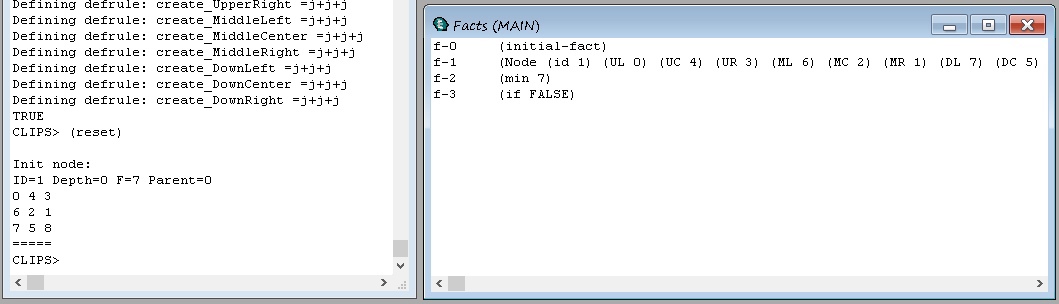


Рис. 13. Инициализация «h1»

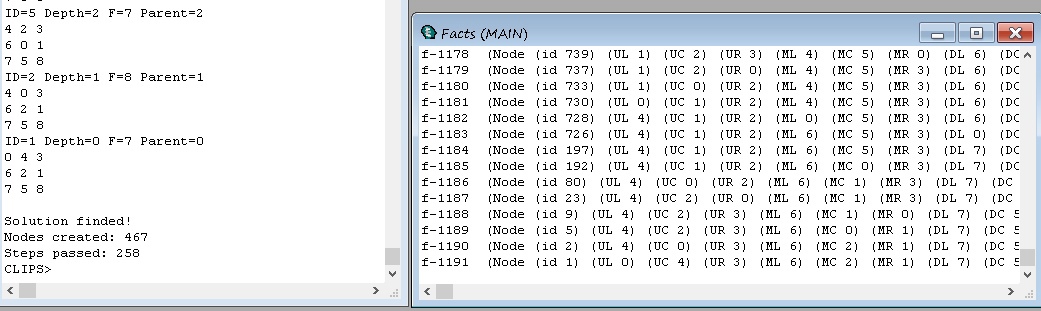


Рис. 14. Результат работы «h1»

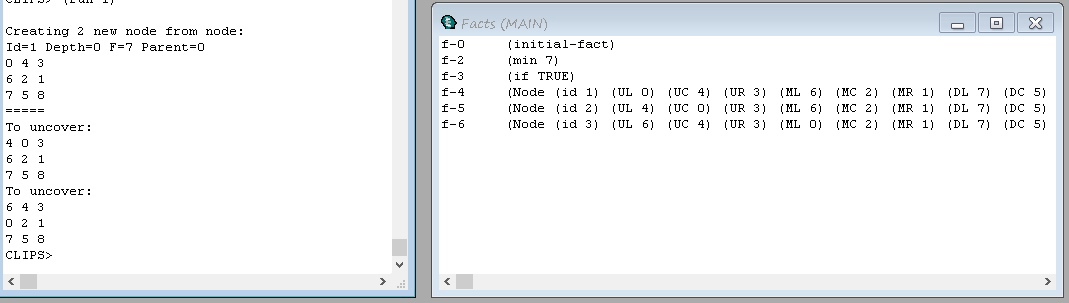


Рис. 15. Результат работы «h1» в пошаговом режиме

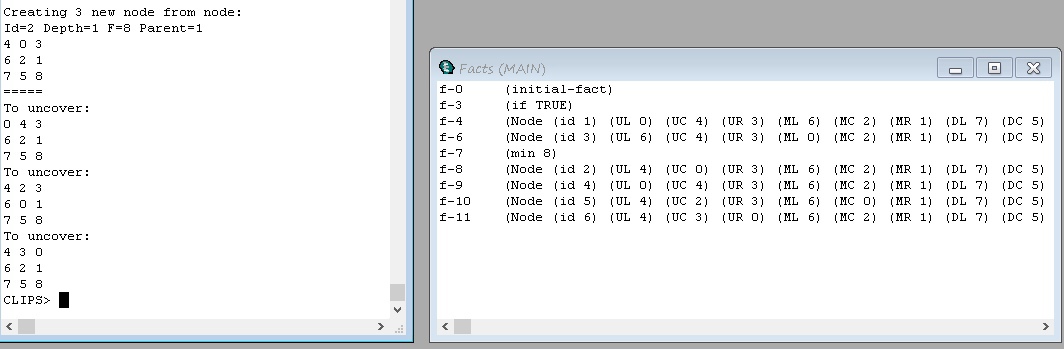


Рис. 16. Результат работы «h1» в пошаговом режиме

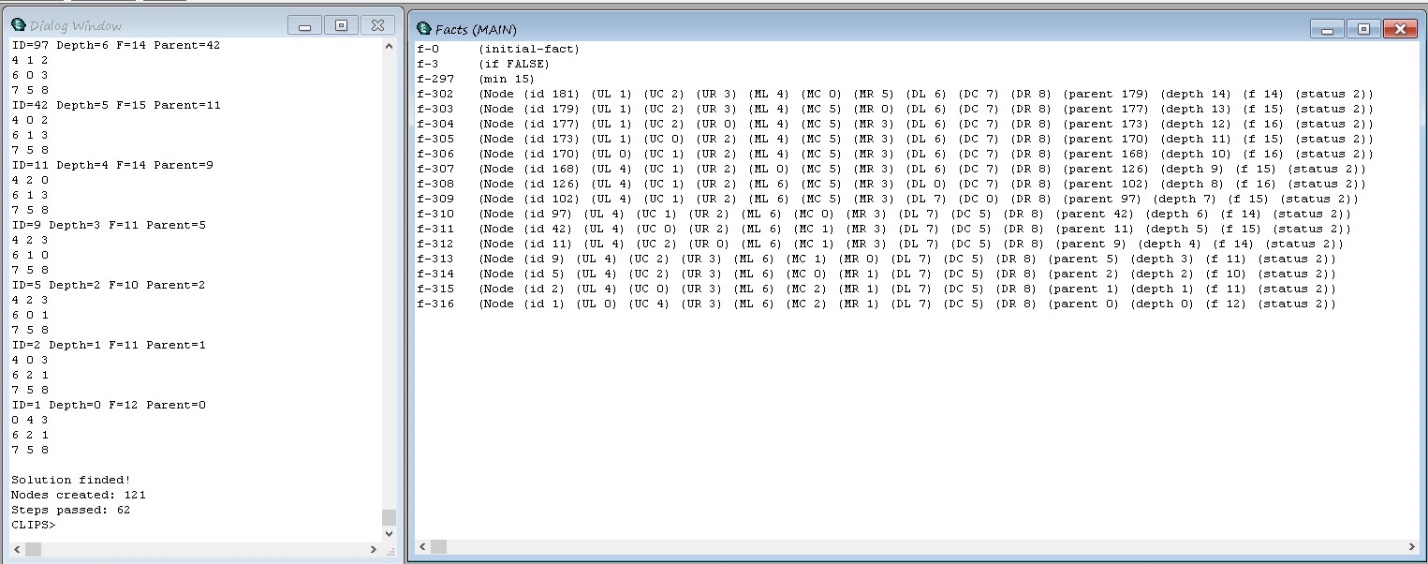


Рис. 17. Результат работы «h2»

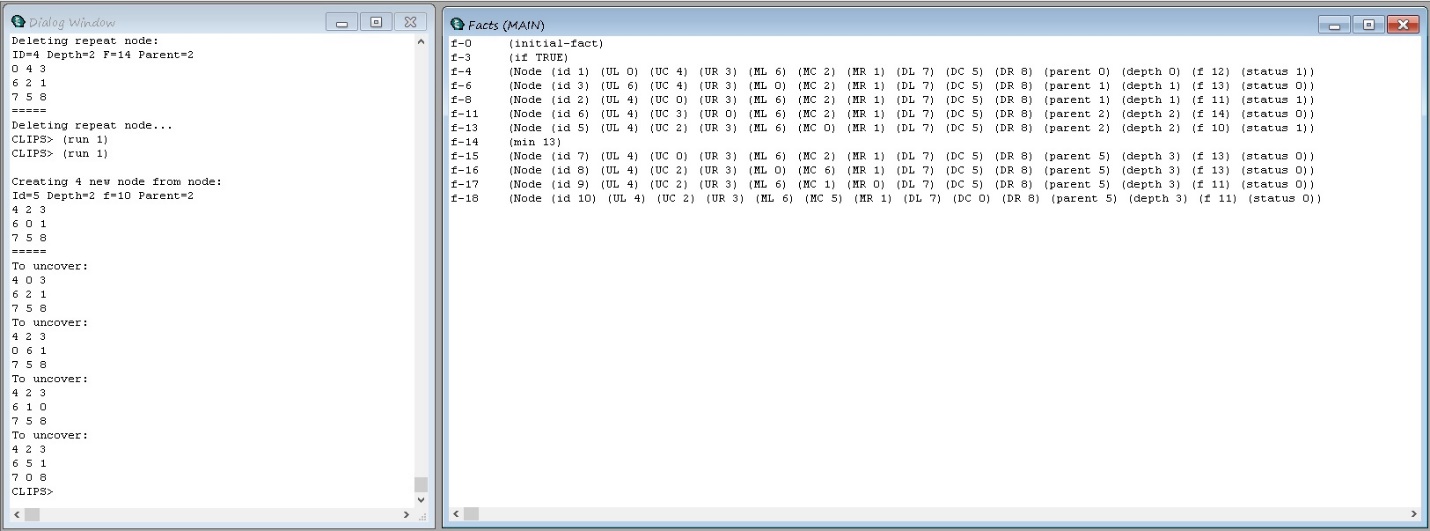


Рис. 18. Результат работы «h2» в пошаговом режиме

**Экспериментальные оценки временной и ёмкостной сложности**



Из экспериментальных замеров видно, что при использовании эвристической функции h2, временная и ёмкостная сложность падает напорядок 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).

**Временная сложность алгоритма**

Каждое правило имеет временную сложность O(b^d). Кроме того, постоянно происходит контроль по определению минимума оценочных функций - O(b^d). Таким образом, временная сложность равна O(b^(2d)).

**Емкостная сложность алгоритма**

Служебная информация (например, глобальная переменная подсчёта id вершин, факт с информацией о наименьшем значении оценочной функции в данном дереве) не зависит от числа входных данных, потому местом, занимаемым ей в памяти, можно пренебречь.

Также в памяти хранятся правила, очевидно, что их кол-во зависит от кол-ва текущих фактов, хранящих информацию о вершинах. Таким образом, правила могут занимать в памяти O(b^d) для каждого дерева. Таким образом, емкостная сложность равна O(b^d).

**Вывод**

// ПЕРЕПИСАТЬ РАСПРЕДЕЛЕНИЕ РОЛЕЙ ВЫВОД ДЛЯ ЛАБОРАТОРНОЙ РАБОТЫ 4

В ходе выполнения лабораторной работы №2 «Методы информированного (эвристического) поиска» были реализованы Алгоритм A\*, а также две эвристические функции h1 (поиск фишек не на своих местах с присваиванием приоритета) и h2 (манхэттенское расстояние), для чего были добавлены дополнительные структуры данных, такие как компаратор и очередь с приоритетом вместо стека, далее была сделана временная оценка сложности алгоритма и стратегий информированного поиска, из чего была подтверждена его эффективность и большая эффективность по сравнению с неинформированным поиском. Таким образом и была выполнена лабораторная работа.

В ходе работы возникли следующие проблемы:

1. Нам был необходим компаратор для сравнения приоритета у вершин. Для реализации компаратора можно было бы выбрать 3 варианта: основываясь на лямбда-функциях, основываясь на классе или основываясь на структуре. Была произведена попытка написать его на лямбда-функциях, но при компиляции и написании возникли ошибки, связанные с системными зависимостями, иными словами, сама архитектура программы не позволила использовать лямбда-функции. Поэтому мы решили использовать структуру, так как её было легче всего объявить, описать и интегрировать в уже имеющийся код.
2. В ходе работы возникла проблема выбора типа данных, который должен поддерживать два типа операций: добавить элемент и извлечь максимум. Выбор пал на очередь с приоритетом («priority\_queue»), где элемент с более высоким приоритетом располагается перед элементом с более низким приоритетом, а если у них он одинаковый, они располагаются исходя из изначальной позиции в структуре. Это было нужно для выполнения алгоритма более оптимальным методом, чтобы сразу основываться на очереди, делегируя сортировку объектов структуре данных.

**Листинг программы**

**Lab4.clp**

;; ======================================

;; Начальная инициализация входных данных

;; ======================================

(defglobal

?\*ID\* = 0

?\*NODES\_CREATED\* = 1

?\*STEPS\_PASSED\* = 0

;; Изначальный граф

?\*init\_UpL\* = 0

?\*init\_UpC\* = 4

?\*init\_UpR\* = 3

?\*init\_MidL\* = 6

?\*init\_MidC\* = 2

?\*init\_MidR\* = 1

?\*init\_DownL\* = 7

?\*init\_DownC\* = 5

?\*init\_DownR\* = 8

;; Конечный путь

?\*result\_UpL\* = 1

?\*result\_UpC\* = 2

?\*result\_UpR\* = 3

?\*result\_MidL\* = 4

?\*result\_MidC\* = 0

?\*result\_MidR\* = 5

?\*result\_DownL\* = 6

?\*result\_DownC\* = 7

?\*result\_DownR\* = 8

;; EF=1:h1 (количество фишек, стоящих не на своем месте); EF=2:h2 (сумма расстояний всех фишек до целевых позиций(манхэттенское расстояние)); DB=1:steps

?\*EvristicFuntion\* = 2

?\*DEBUG\* = 0

)

;; ======================================

;; Шаблон вершины

;; ======================================

(deftemplate Node

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

(slot UL (type NUMBER))

(slot UC (type NUMBER))

(slot UR (type NUMBER))

(slot ML (type NUMBER))

(slot MC (type NUMBER))

(slot MR (type NUMBER))

(slot DL (type NUMBER))

(slot DC (type NUMBER))

(slot DR (type NUMBER))

(slot parent (type NUMBER))

(slot depth (type NUMBER))

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

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

)

;; ======================================

;; Сумма Манхэттенских расстояний фишек от из целевой позиции (для эвристической функции h2)

;; ======================================

(deffunction manhattanFunc(?v ?i ?j)

(if (= ?v ?\*result\_UpL\*) then (bind ?i\_g 0))

(if (= ?v ?\*result\_UpL\*) then (bind ?j\_g 0))

(if (= ?v ?\*result\_UpC\*) then (bind ?i\_g 0))

(if (= ?v ?\*result\_UpC\*) then (bind ?j\_g 1))

(if (= ?v ?\*result\_UpR\*) then (bind ?i\_g 0))

(if (= ?v ?\*result\_UpR\*) then (bind ?j\_g 2))

(if (= ?v ?\*result\_MidL\*) then (bind ?i\_g 1))

(if (= ?v ?\*result\_MidL\*) then (bind ?j\_g 0))

(if (= ?v ?\*result\_MidC\*) then (bind ?i\_g 1))

(if (= ?v ?\*result\_MidC\*) then (bind ?j\_g 1))

(if (= ?v ?\*result\_MidR\*) then (bind ?i\_g 1))

(if (= ?v ?\*result\_MidR\*) then (bind ?j\_g 2))

(if (= ?v ?\*result\_DownL\*) then (bind ?i\_g 2))

(if (= ?v ?\*result\_DownL\*) then (bind ?j\_g 0))

(if (= ?v ?\*result\_DownC\*) then (bind ?i\_g 2))

(if (= ?v ?\*result\_DownC\*) then (bind ?j\_g 1))

(if (= ?v ?\*result\_DownR\*) then (bind ?i\_g 2))

(if (= ?v ?\*result\_DownR\*) then (bind ?j\_g 2))

(return (+ (abs (- ?i ?i\_g)) (abs (- ?j ?j\_g))))

)

;; ======================================

;; Евристика h1

;; ======================================

(deffunction h1(?cur\_UL ?cur\_UC ?cur\_UR

?cur\_ML ?cur\_MC ?cur\_MR

?cur\_DL ?cur\_DC ?cur\_DR)

(bind ?a 0)

(if (not (= ?cur\_UL ?\*result\_UpL\*)) then (bind ?a (+ ?a 1)))

(if (not (= ?cur\_UC ?\*result\_UpC\*)) then (bind ?a (+ ?a 1)))

(if (not (= ?cur\_UR ?\*result\_UpR\*)) then (bind ?a (+ ?a 1)))

(if (not (= ?cur\_ML ?\*result\_MidL\*)) then (bind ?a (+ ?a 1)))

(if (not (= ?cur\_MC ?\*result\_MidC\*)) then (bind ?a (+ ?a 1)))

(if (not (= ?cur\_MR ?\*result\_MidR\*)) then (bind ?a (+ ?a 1)))

(if (not (= ?cur\_DL ?\*result\_DownL\*)) then (bind ?a (+ ?a 1)))

(if (not (= ?cur\_DC ?\*result\_DownC\*)) then (bind ?a (+ ?a 1)))

(if (not (= ?cur\_DR ?\*result\_DownR\*)) then (bind ?a (+ ?a 1)))

(return ?a)

)

;; ======================================

;; Евристика h2

;; ======================================

(deffunction h2(?cur\_UL ?cur\_UC ?cur\_UR

?cur\_ML ?cur\_MC ?cur\_MR

?cur\_DL ?cur\_DC ?cur\_DR)

(bind ?a 0)

(bind ?a (+ ?a (manhattanFunc ?cur\_UL 0 0)))

(bind ?a (+ ?a (manhattanFunc ?cur\_UC 0 1)))

(bind ?a (+ ?a (manhattanFunc ?cur\_UR 0 2)))

(bind ?a (+ ?a (manhattanFunc ?cur\_ML 1 0)))

(bind ?a (+ ?a (manhattanFunc ?cur\_MC 1 1)))

(bind ?a (+ ?a (manhattanFunc ?cur\_MR 1 2)))

(bind ?a (+ ?a (manhattanFunc ?cur\_DL 2 0)))

(bind ?a (+ ?a (manhattanFunc ?cur\_DC 2 1)))

(bind ?a (+ ?a (manhattanFunc ?cur\_DR 2 2)))

(return ?a)

)

;; ======================================

;; Функция расчета оценочной стоимости евристики /приоритета (h1 и h2 соответственно)

;; ======================================

(deffunction getF(?depth

?cur\_UL ?cur\_UC ?cur\_UR

?cur\_ML ?cur\_MC ?cur\_MR

?cur\_DL ?cur\_DC ?cur\_DR)

(bind ?a ?depth)

(if (= ?\*EvristicFuntion\* 1) then (bind ?a (+ ?a (h1 ?cur\_UL ?cur\_UC ?cur\_UR ?cur\_ML ?cur\_MC ?cur\_MR ?cur\_DL ?cur\_DC ?cur\_DR))))

(if (= ?\*EvristicFuntion\* 2) then (bind ?a (+ ?a (h2 ?cur\_UL ?cur\_UC ?cur\_UR ?cur\_ML ?cur\_MC ?cur\_MR ?cur\_DL ?cur\_DC ?cur\_DR))))

(return ?a)

)

;; ======================================

;; Функция присваивания ID узлу

;; ======================================

(deffunction setID()

(bind ?\*ID\* (+ ?\*ID\* 1))

(return ?\*ID\*)

)

;; ======================================

;; Инициализация начальной вершины (reset)

;; ======================================

(deffacts initial

(Node (id (setID))

(UL ?\*init\_UpL\*) (UC ?\*init\_UpC\*) (UR ?\*init\_UpR\*)

(ML ?\*init\_MidL\*) (MC ?\*init\_MidC\*) (MR ?\*init\_MidR\*)

(DL ?\*init\_DownL\*) (DC ?\*init\_DownC\*) (DR ?\*init\_DownR\*)

(depth 0) (parent 0)

(f (getF 0 ?\*init\_UpL\* ?\*init\_UpC\* ?\*init\_UpR\* ?\*init\_MidL\* ?\*init\_MidC\* ?\*init\_MidR\* ?\*init\_DownL\* ?\*init\_DownC\* ?\*init\_DownR\*))

)

(min (getF 0 ?\*init\_UpL\* ?\*init\_UpC\* ?\*init\_UpR\* ?\*init\_MidL\* ?\*init\_MidC\* ?\*init\_MidR\* ?\*init\_DownL\* ?\*init\_DownC\* ?\*init\_DownR\*))

(if (= ?\*DEBUG\* 1)

(printout t crlf "Init node: " crlf

"ID=1 " "Depth=0 " "F=" (getF 0 ?\*init\_UpL\* ?\*init\_UpC\* ?\*init\_UpR\* ?\*init\_MidL\* ?\*init\_MidC\* ?\*init\_MidR\* ?\*init\_DownL\* ?\*init\_DownC\* ?\*init\_DownR\*) " Parent=0 " crlf

?\*init\_UpL\* " " ?\*init\_UpC\* " " ?\*init\_UpR\* crlf

?\*init\_MidL\* " " ?\*init\_MidC\* " " ?\*init\_MidR\* crlf

?\*init\_DownL\* " " ?\*init\_DownC\* " " ?\*init\_DownR\* crlf

"====="crlf

)

)

)

;; ======================================

;; Вывод найденного решения

;; ======================================

(defrule result

(declare (salience 500))

?container <- (Node (id ?v\_id)

(UL ?v\_UL) (UC ?v\_UC) (UR ?v\_UR)

(ML ?v\_ML) (MC ?v\_MC) (MR ?v\_MR)

(DL ?v\_DL) (DC ?v\_DC) (DR ?v\_DR)

(depth ?v\_g) (status ~2) (parent ?v\_parent) (f ?v\_f)

)

(test (= ?v\_UL ?\*result\_UpL\*))

(test (= ?v\_UC ?\*result\_UpC\*))

(test (= ?v\_UR ?\*result\_UpR\*))

(test (= ?v\_ML ?\*result\_MidL\*))

(test (= ?v\_MC ?\*result\_MidC\*))

(test (= ?v\_MR ?\*result\_MidR\*))

(test (= ?v\_DL ?\*result\_DownL\*))

(test (= ?v\_DC ?\*result\_DownC\*))

(test (= ?v\_DR ?\*result\_DownR\*))

=>

(modify ?container(status 2))

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

"Result path: " crlf

"ID=" ?v\_id " Depth=" ?v\_g " F=" ?v\_f " Parent=" ?v\_parent crlf

?v\_UL " " ?v\_UC " " ?v\_UR crlf

?v\_ML " " ?v\_MC " " ?v\_MR crlf

?v\_DL " " ?v\_DC " " ?v\_DR crlf

)

)

;; ======================================

;; Случай если решения не найдено

;; ======================================

(defrule stop\_if\_no\_solution

(declare (salience 200))

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

=>

(halt)

(printout t crlf "Solution not finded." crlf)

)

;; ======================================

;; Случай если решения найдено, остановливаем программу

;; ======================================

(defrule stop\_if\_solution\_finded

(declare (salience 200))

(Node(status 2))

=>

(halt)

(printout t crlf "Solution finded! " crlf)

(printout t "Nodes created: " ?\*NODES\_CREATED\* crlf)

(printout t "Steps passed: " ?\*STEPS\_PASSED\* crlf)

)

;; ======================================

;; Случай если не было найдено минимума

;; ======================================

(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)))

)

;; ======================================

;; Определение текущего минимума ЦФ

;; ======================================

(defrule find\_min

(declare (salience 150))

?f\_addr\_min <- (min ?min)

(Node (f ?F&:(< ?F ?min)) (status 0))

=>

(retract ?f\_addr\_min)

(assert (min ?F))

)

;; ======================================

;; Функция удаления повторяющихся вершин

;; ======================================

(defrule remove\_repeats

(declare (salience 1000)); максимальный приоритет

?f\_addr\_1 <-(Node (id ?v\_id\_1)

(UL ?v\_UL\_1) (UC ?v\_UC\_1) (UR ?v\_UR\_1)

(ML ?v\_ML\_1) (MC ?v\_MC\_1) (MR ?v\_MR\_1)

(DL ?v\_DL\_1) (DC ?v\_DC\_1) (DR ?v\_DR\_1)

(depth ?v\_g\_1) (status 1) (parent ?v\_parent\_1) (f ?v\_f\_1)

)

?f\_addr\_2 <-(Node (id ?v\_id\_2&~?v\_id\_1)

(UL ?v\_UL\_2&:(= ?v\_UL\_1 ?v\_UL\_2)) (UC ?v\_UC\_2&:(= ?v\_UC\_1 ?v\_UC\_2)) (UR ?v\_UR\_2&:(= ?v\_UR\_1 ?v\_UR\_2))

(ML ?v\_ML\_2&:(= ?v\_ML\_1 ?v\_ML\_2)) (MC ?v\_MC\_2&:(= ?v\_MC\_1 ?v\_MC\_2)) (MR ?v\_MR\_2&:(= ?v\_MR\_1 ?v\_MR\_2))

(DL ?v\_DL\_2&:(= ?v\_DL\_1 ?v\_DL\_2)) (DC ?v\_DC\_2&:(= ?v\_DC\_1 ?v\_DC\_2)) (DR ?v\_DR\_2&:(= ?v\_DR\_1 ?v\_DR\_2))

(depth ?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 " Depth=" ?v\_g\_2 " F=" ?v\_f\_2 " Parent=" ?v\_parent\_2 crlf

?v\_UL\_2 " " ?v\_UC\_2 " " ?v\_UR\_2 crlf

?v\_ML\_2 " " ?v\_MC\_2 " " ?v\_MR\_2 crlf

?v\_DL\_2 " " ?v\_DC\_2 " " ?v\_DR\_2 crlf

"=====" crlf

)

)

(if (<= ?v\_f\_1 ?v\_f\_2) then

(retract ?f\_addr\_2)

(if (= ?\*DEBUG\* 1) then (printout t "Deleting repeat node..." crlf))

else

(modify ?f\_addr\_1 (parent ?v\_parent\_2) (depth ?v\_g\_2) (f ?v\_f\_2))

(retract ?f\_addr\_2)

(if (= ?\*DEBUG\* 1) then (printout t "Deleting and refreshing prev repeat" crlf))

)

(bind ?\*NODES\_CREATED\* (- ?\*NODES\_CREATED\* 1))

)

;; ======================================

;; Функция вывода результирующего конечного путя

;; ======================================

(defrule show\_answer

(declare (salience 500))

(Node (id ?v\_id) (status 2) (parent ?v\_pid))

?container <- (Node(UL ?v\_UL) (UC ?v\_UC) (UR ?v\_UR)

(ML ?v\_ML) (MC ?v\_MC) (MR ?v\_MR)

(DL ?v\_DL) (DC ?v\_DC) (DR ?v\_DR)

(id ?v\_pid) (status ~2) (parent ?v\_parent) (depth ?v\_g) (f ?v\_f))

=>

(modify ?container(status 2))

(printout t "ID=" ?v\_pid " Depth=" ?v\_g " F=" ?v\_f " Parent=" ?v\_parent crlf

?v\_UL " " ?v\_UC " " ?v\_UR crlf

?v\_ML " " ?v\_MC " " ?v\_MR crlf

?v\_DL " " ?v\_DC " " ?v\_DR crlf

)

)

;; ======================================

;; Функция удаления вершин у которых статус не соотвествует 2 (решению)

;; ======================================

(defrule delete\_not\_answer

(declare (salience 400))

(Node(status 2))

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

=>

(retract ?container);

)

;; ======================================

;; Создание новых вершин/всевозможное перемещение "0"

;; ======================================

(defrule create\_UpperLeft

(declare (salience 100))

?f\_addr\_min <- (min ?min)

?f\_addr\_node <- (Node (id ?v\_id)

(UL 0) (UC ?v\_UC) (UR ?v\_UR)

(ML ?v\_ML) (MC ?v\_MC) (MR ?v\_MR)

(DL ?v\_DL) (DC ?v\_DC) (DR ?v\_DR)

(depth ?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 " Depth=" ?v\_g " F=" ?v\_f " Parent=" ?v\_parent crlf

(fact-slot-value ?f\_addr\_node UL) " " (fact-slot-value ?f\_addr\_node UC) " " (fact-slot-value ?f\_addr\_node UR) crlf

(fact-slot-value ?f\_addr\_node ML) " " (fact-slot-value ?f\_addr\_node MC) " " (fact-slot-value ?f\_addr\_node MR) crlf

(fact-slot-value ?f\_addr\_node DL) " " (fact-slot-value ?f\_addr\_node DC) " " (fact-slot-value ?f\_addr\_node DR) crlf

"====="crlf

)

)

(modify ?f\_addr\_node(status 1))

(bind ?a1 (getF (+ ?v\_g 1) ?v\_UC 0 ?v\_UR ?v\_ML ?v\_MC ?v\_MR ?v\_DL ?v\_DC ?v\_DR))

(assert (Node (id (setID))

(UL ?v\_UC) (UC 0 ) (UR ?v\_UR)

(ML ?v\_ML) (MC ?v\_MC) (MR ?v\_MR)

(DL ?v\_DL) (DC ?v\_DC) (DR ?v\_DR)

(depth (+ ?v\_g 1)) (status 0) (parent ?v\_id) (f ?a1)

)

)

(if (= ?\*DEBUG\* 1) then (printout t "To uncover:" crlf

?v\_UC " 0 " ?v\_UR crlf

?v\_ML " " ?v\_MC " " ?v\_MR crlf

?v\_DL " " ?v\_DC " " ?v\_DR crlf

)

)

(bind ?a2 (getF (+ ?v\_g 1) ?v\_ML ?v\_UC ?v\_UR 0 ?v\_MC ?v\_MR ?v\_DL ?v\_DC ?v\_DR))

(assert (Node (id (setID))

(UL ?v\_ML) (UC ?v\_UC) (UR ?v\_UR)

(ML 0) (MC ?v\_MC) (MR ?v\_MR)

(DL ?v\_DL) (DC ?v\_DC) (DR ?v\_DR)

(depth (+ ?v\_g 1)) (status 0) (parent ?v\_id) (f ?a2)

)

)

(if (= ?\*DEBUG\* 1) then (printout t "To uncover:" crlf

?v\_ML " " ?v\_UC " " ?v\_UR crlf

"0 " ?v\_MC " " ?v\_MR crlf

?v\_DL " " ?v\_DC " " ?v\_DR crlf

)

)

(bind ?\*NODES\_CREATED\* (+ ?\*NODES\_CREATED\* 2))

(bind ?\*STEPS\_PASSED\* (+ ?\*STEPS\_PASSED\* 1))

)

(defrule create\_UpperCenter

(declare (salience 100))

?f\_addr\_min <- (min ?min)

?f\_addr\_node <- (Node (id ?v\_id)

(UL ?v\_UL) (UC 0) (UR ?v\_UR)

(ML ?v\_ML) (MC ?v\_MC) (MR ?v\_MR)

(DL ?v\_DL) (DC ?v\_DC) (DR ?v\_DR)

(depth ?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 " Depth=" ?v\_g " F=" ?v\_f " Parent=" ?v\_parent crlf

(fact-slot-value ?f\_addr\_node UL) " " (fact-slot-value ?f\_addr\_node UC) " " (fact-slot-value ?f\_addr\_node UR) crlf

(fact-slot-value ?f\_addr\_node ML) " " (fact-slot-value ?f\_addr\_node MC) " " (fact-slot-value ?f\_addr\_node MR) crlf

(fact-slot-value ?f\_addr\_node DL) " " (fact-slot-value ?f\_addr\_node DC) " " (fact-slot-value ?f\_addr\_node DR) crlf

"====="crlf

)

)

(modify ?f\_addr\_node(status 1))

(bind ?a1 (getF (+ ?v\_g 1) 0 ?v\_UL ?v\_UR ?v\_ML ?v\_MC ?v\_MR ?v\_DL ?v\_DC ?v\_DR))

(assert (Node (id (setID))

(UL 0) (UC ?v\_UL) (UR ?v\_UR)

(ML ?v\_ML) (MC ?v\_MC) (MR ?v\_MR)

(DL ?v\_DL) (DC ?v\_DC) (DR ?v\_DR)

(depth (+ ?v\_g 1)) (status 0) (parent ?v\_id) (f ?a1)

)

)

(if (= ?\*DEBUG\* 1) then (printout t "To uncover:" crlf

"0 " ?v\_UL " " ?v\_UR crlf

?v\_ML " " ?v\_MC " " ?v\_MR crlf

?v\_DL " " ?v\_DC " " ?v\_DR crlf

)

)

(bind ?a2 (getF (+ ?v\_g 1) ?v\_UL ?v\_MC ?v\_UR ?v\_ML 0 ?v\_MR ?v\_DL ?v\_DC ?v\_DR))

(assert (Node (id (setID))

(UL ?v\_UL) (UC ?v\_MC) (UR ?v\_UR)

(ML ?v\_ML) (MC 0) (MR ?v\_MR)

(DL ?v\_DL) (DC ?v\_DC) (DR ?v\_DR)

(depth (+ ?v\_g 1)) (status 0) (parent ?v\_id) (f ?a2)

)

)

(if (= ?\*DEBUG\* 1) then (printout t "To uncover:" crlf

?v\_UL " " ?v\_MC " " ?v\_UR crlf

?v\_ML " 0 " ?v\_MR crlf

?v\_DL " " ?v\_DC " " ?v\_DR crlf

)

)

(bind ?a3 (getF (+ ?v\_g 1) ?v\_UL ?v\_UR 0 ?v\_ML ?v\_MC ?v\_MR ?v\_DL ?v\_DC ?v\_DR))

(assert (Node (id (setID))

(UL ?v\_UL) (UC ?v\_UR) (UR 0)

(ML ?v\_ML) (MC ?v\_MC) (MR ?v\_MR)

(DL ?v\_DL) (DC ?v\_DC) (DR ?v\_DR)

(depth (+ ?v\_g 1)) (status 0) (parent ?v\_id) (f ?a3)

)

)

(if (= ?\*DEBUG\* 1) then (printout t "To uncover:" crlf

?v\_UL " " ?v\_UR " 0" crlf

?v\_ML " " ?v\_MC " " ?v\_MR crlf

?v\_DL " " ?v\_DC " " ?v\_DR crlf

)

)

(bind ?\*NODES\_CREATED\* (+ ?\*NODES\_CREATED\* 3))

(bind ?\*STEPS\_PASSED\* (+ ?\*STEPS\_PASSED\* 1))

)

(defrule create\_UpperRight

(declare (salience 100))

?f\_addr\_min <- (min ?min)

?f\_addr\_node <- (Node (id ?v\_id)

(UL ?v\_UL) (UC ?v\_UC) (UR 0)

(ML ?v\_ML) (MC ?v\_MC) (MR ?v\_MR)

(DL ?v\_DL) (DC ?v\_DC) (DR ?v\_DR)

(depth ?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 " Depth=" ?v\_g " F=" ?v\_f " Parent=" ?v\_parent crlf

(fact-slot-value ?f\_addr\_node UL) " " (fact-slot-value ?f\_addr\_node UC) " " (fact-slot-value ?f\_addr\_node UR) crlf

(fact-slot-value ?f\_addr\_node ML) " " (fact-slot-value ?f\_addr\_node MC) " " (fact-slot-value ?f\_addr\_node MR) crlf

(fact-slot-value ?f\_addr\_node DL) " " (fact-slot-value ?f\_addr\_node DC) " " (fact-slot-value ?f\_addr\_node DR) crlf

"====="crlf

)

)

(modify ?f\_addr\_node(status 1))

(bind ?a1 (getF (+ ?v\_g 1) ?v\_UL 0 ?v\_UC ?v\_ML ?v\_MC ?v\_MR ?v\_DL ?v\_DC ?v\_DR))

(assert (Node (id (setID))

(UL ?v\_UL) (UC 0) (UR ?v\_UC)

(ML ?v\_ML) (MC ?v\_MC) (MR ?v\_MR)

(DL ?v\_DL) (DC ?v\_DC) (DR ?v\_DR)

(depth (+ ?v\_g 1)) (status 0) (parent ?v\_id) (f ?a1)

)

)

(if (= ?\*DEBUG\* 1) then (printout t "To uncover:" crlf

?v\_UL " 0 " ?v\_UC crlf

?v\_ML " " ?v\_MC " " ?v\_MR crlf

?v\_DL " " ?v\_DC " " ?v\_DR crlf

)

)

(bind ?a2 (getF (+ ?v\_g 1) ?v\_UL ?v\_UC ?v\_MR ?v\_ML ?v\_MC 0 ?v\_DL ?v\_DC ?v\_DR))

(assert (Node (id (setID))

(UL ?v\_UL) (UC ?v\_UC) (UR ?v\_MR)

(ML ?v\_ML) (MC ?v\_MC) (MR 0)

(DL ?v\_DL) (DC ?v\_DC) (DR ?v\_DR)

(depth (+ ?v\_g 1)) (status 0) (parent ?v\_id) (f ?a2)

)

)

(if (= ?\*DEBUG\* 1) then (printout t "To uncover:" crlf

?v\_UL " " ?v\_UC " " ?v\_MR crlf

?v\_ML " " ?v\_MC " 0" crlf

?v\_DL " " ?v\_DC " " ?v\_DR crlf

)

)

(bind ?\*NODES\_CREATED\* (+ ?\*NODES\_CREATED\* 2))

(bind ?\*STEPS\_PASSED\* (+ ?\*STEPS\_PASSED\* 1))

)

(defrule create\_MiddleLeft

(declare (salience 100))

?f\_addr\_min <- (min ?min)

?f\_addr\_node <- (Node (id ?v\_id)

(UL ?v\_UL) (UC ?v\_UC) (UR ?v\_UR)

(ML 0) (MC ?v\_MC) (MR ?v\_MR)

(DL ?v\_DL) (DC ?v\_DC) (DR ?v\_DR)

(depth ?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 " Depth=" ?v\_g " F=" ?v\_f " Parent=" ?v\_parent crlf

(fact-slot-value ?f\_addr\_node UL) " " (fact-slot-value ?f\_addr\_node UC) " " (fact-slot-value ?f\_addr\_node UR) crlf

(fact-slot-value ?f\_addr\_node ML) " " (fact-slot-value ?f\_addr\_node MC) " " (fact-slot-value ?f\_addr\_node MR) crlf

(fact-slot-value ?f\_addr\_node DL) " " (fact-slot-value ?f\_addr\_node DC) " " (fact-slot-value ?f\_addr\_node DR) crlf

"====="crlf

)

)

(modify ?f\_addr\_node(status 1))

(bind ?a1 (getF (+ ?v\_g 1) 0 ?v\_UC ?v\_UR ?v\_UL ?v\_MC ?v\_MR ?v\_DL ?v\_DC ?v\_DR))

(assert (Node (id (setID))

(UL 0) (UC ?v\_UC) (UR ?v\_UR)

(ML ?v\_UL) (MC ?v\_MC) (MR ?v\_MR)

(DL ?v\_DL) (DC ?v\_DC) (DR ?v\_DR)

(depth (+ ?v\_g 1)) (status 0) (parent ?v\_id) (f ?a1)

)

)

(if (= ?\*DEBUG\* 1) then (printout t "To uncover:" crlf

"0 " ?v\_UC " " ?v\_UR crlf

?v\_UL " " ?v\_MC " " ?v\_MR crlf

?v\_DL " " ?v\_DC " " ?v\_DR crlf

)

)

(bind ?a2 (getF (+ ?v\_g 1) ?v\_UL ?v\_UC ?v\_UR ?v\_MC 0 ?v\_MR ?v\_DL ?v\_DC ?v\_DR))

(assert (Node (id (setID))

(UL ?v\_UL) (UC ?v\_UC) (UR ?v\_UR)

(ML ?v\_MC) (MC 0) (MR ?v\_MR)

(DL ?v\_DL) (DC ?v\_DC) (DR ?v\_DR)

(depth (+ ?v\_g 1)) (status 0) (parent ?v\_id) (f ?a2)

)

)

(if (= ?\*DEBUG\* 1) then (printout t "To uncover:" crlf

?v\_UL " " ?v\_UC " " ?v\_UR crlf

?v\_MC " 0 " ?v\_MR crlf

?v\_DL " " ?v\_DC " " ?v\_DR crlf

)

)

(bind ?a3 (getF (+ ?v\_g 1) ?v\_UL ?v\_UC ?v\_UR ?v\_DL ?v\_MC ?v\_MR 0 ?v\_DC ?v\_DR))

(assert (Node (id (setID))

(UL ?v\_UL) (UC ?v\_UC) (UR ?v\_UR)

(ML ?v\_DL) (MC ?v\_MC) (MR ?v\_MR)

(DL 0) (DC ?v\_DC) (DR ?v\_DR)

(depth (+ ?v\_g 1)) (status 0) (parent ?v\_id) (f ?a3)

)

)

(if (= ?\*DEBUG\* 1) then (printout t "To uncover:" crlf

?v\_UL " " ?v\_UC " " ?v\_UR crlf

?v\_DL " " ?v\_MC " " ?v\_MR crlf

"0 " ?v\_DC " " ?v\_DR crlf

)

)

(bind ?\*NODES\_CREATED\* (+ ?\*NODES\_CREATED\* 3))

(bind ?\*STEPS\_PASSED\* (+ ?\*STEPS\_PASSED\* 1))

)

(defrule create\_MiddleCenter

(declare (salience 100))

?f\_addr\_min <- (min ?min)

?f\_addr\_node <- (Node (id ?v\_id)

(UL ?v\_UL) (UC ?v\_UC) (UR ?v\_UR)

(ML ?v\_ML) (MC 0) (MR ?v\_MR)

(DL ?v\_DL) (DC ?v\_DC) (DR ?v\_DR)

(depth ?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 " Depth=" ?v\_g " f=" ?v\_f " Parent=" ?v\_parent crlf

(fact-slot-value ?f\_addr\_node UL) " " (fact-slot-value ?f\_addr\_node UC) " " (fact-slot-value ?f\_addr\_node UR) crlf

(fact-slot-value ?f\_addr\_node ML) " " (fact-slot-value ?f\_addr\_node MC) " " (fact-slot-value ?f\_addr\_node MR) crlf

(fact-slot-value ?f\_addr\_node DL) " " (fact-slot-value ?f\_addr\_node DC) " " (fact-slot-value ?f\_addr\_node DR) crlf

"====="crlf

)

)

(modify ?f\_addr\_node(status 1))

(bind ?a1 (getF (+ ?v\_g 1) ?v\_UL 0 ?v\_UR ?v\_ML ?v\_UC ?v\_MR ?v\_DL ?v\_DC ?v\_DR))

(retract ?f\_addr\_min)

(assert (min ?a1))

(assert (Node (id (setID))

(UL ?v\_UL) (UC 0) (UR ?v\_UR)

(ML ?v\_ML) (MC ?v\_UC) (MR ?v\_MR)

(DL ?v\_DL) (DC ?v\_DC) (DR ?v\_DR)

(depth (+ ?v\_g 1)) (status 0) (parent ?v\_id) (f ?a1)

)

)

(if (= ?\*DEBUG\* 1) then (printout t "To uncover:" crlf

?v\_UL " 0 " ?v\_UR crlf

?v\_ML " " ?v\_UC " " ?v\_MR crlf

?v\_DL " " ?v\_DC " " ?v\_DR crlf

)

)

(bind ?a2 (getF (+ ?v\_g 1) ?v\_UL ?v\_UC ?v\_UR 0 ?v\_ML ?v\_MR ?v\_DL ?v\_DC ?v\_DR))

(assert (Node (id (setID))

(UL ?v\_UL) (UC ?v\_UC) (UR ?v\_UR)

(ML 0) (MC ?v\_ML) (MR ?v\_MR)

(DL ?v\_DL) (DC ?v\_DC) (DR ?v\_DR)

(depth (+ ?v\_g 1)) (status 0) (parent ?v\_id) (f ?a2)

)

)

(if (= ?\*DEBUG\* 1) then (printout t "To uncover:" crlf

?v\_UL " " ?v\_UC " " ?v\_UR crlf

"0 " ?v\_ML " " ?v\_MR crlf

?v\_DL " " ?v\_DC " " ?v\_DR crlf

)

)

(bind ?a3 (getF (+ ?v\_g 1) ?v\_UL ?v\_UC ?v\_UR ?v\_ML ?v\_MR 0 ?v\_DL ?v\_DC ?v\_DR))

(assert (Node (id (setID))

(UL ?v\_UL) (UC ?v\_UC) (UR ?v\_UR)

(ML ?v\_ML) (MC ?v\_MR) (MR 0)

(DL ?v\_DL) (DC ?v\_DC) (DR ?v\_DR)

(depth (+ ?v\_g 1)) (status 0) (parent ?v\_id) (f ?a3)

)

)

(if (= ?\*DEBUG\* 1) then (printout t "To uncover:" crlf

?v\_UL " " ?v\_UC " " ?v\_UR crlf

?v\_ML " " ?v\_MR " 0" crlf

?v\_DL " " ?v\_DC " " ?v\_DR crlf

)

)

(bind ?a4 (getF (+ ?v\_g 1) ?v\_UL ?v\_UC ?v\_UR ?v\_ML ?v\_DC ?v\_MR ?v\_DL 0 ?v\_DR))

(assert (Node (id (setID))

(UL ?v\_UL) (UC ?v\_UC) (UR ?v\_UR)

(ML ?v\_ML) (MC ?v\_DC) (MR ?v\_MR)

(DL ?v\_DL) (DC 0) (DR ?v\_DR)

(depth (+ ?v\_g 1)) (status 0) (parent ?v\_id) (f ?a4)

)

)

(if (= ?\*DEBUG\* 1) then (printout t "To uncover:" crlf

?v\_UL " " ?v\_UC " " ?v\_UR crlf

?v\_ML " " ?v\_DC " " ?v\_MR crlf

?v\_DL " 0 " ?v\_DR crlf

)

)

(bind ?\*NODES\_CREATED\* (+ ?\*NODES\_CREATED\* 4))

(bind ?\*STEPS\_PASSED\* (+ ?\*STEPS\_PASSED\* 1))

)

(defrule create\_MiddleRight

(declare (salience 100))

?f\_addr\_min <- (min ?min)

?f\_addr\_node <- (Node (id ?v\_id)

(UL ?v\_UL) (UC ?v\_UC) (UR ?v\_UR)

(ML ?v\_ML) (MC ?v\_MC) (MR 0)

(DL ?v\_DL) (DC ?v\_DC) (DR ?v\_DR)

(depth ?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 " Depth=" ?v\_g " f=" ?v\_f " Parent=" ?v\_parent crlf

(fact-slot-value ?f\_addr\_node UL) " " (fact-slot-value ?f\_addr\_node UC) " " (fact-slot-value ?f\_addr\_node UR) crlf

(fact-slot-value ?f\_addr\_node ML) " " (fact-slot-value ?f\_addr\_node MC) " " (fact-slot-value ?f\_addr\_node MR) crlf

(fact-slot-value ?f\_addr\_node DL) " " (fact-slot-value ?f\_addr\_node DC) " " (fact-slot-value ?f\_addr\_node DR) crlf

"====="crlf

)

)

(modify ?f\_addr\_node(status 1))

(bind ?a1 (getF (+ ?v\_g 1) ?v\_UL ?v\_UC 0 ?v\_ML ?v\_MC ?v\_UR ?v\_DL ?v\_DC ?v\_DR))

(assert (Node (id (setID))

(UL ?v\_UL) (UC ?v\_UC) (UR 0)

(ML ?v\_ML) (MC ?v\_MC) (MR ?v\_UR)

(DL ?v\_DL) (DC ?v\_DC) (DR ?v\_DR)

(depth (+ ?v\_g 1)) (status 0) (parent ?v\_id) (f ?a1)

)

)

(if (= ?\*DEBUG\* 1) then (printout t "To uncover:" crlf

?v\_UL " " ?v\_UC " 0" crlf

?v\_ML " " ?v\_MC " " ?v\_UR crlf

?v\_DL " " ?v\_DC " " ?v\_DR crlf

)

)

(bind ?a2 (getF (+ ?v\_g 1) ?v\_UL ?v\_UC ?v\_UR ?v\_ML 0 ?v\_MC ?v\_DL ?v\_DC ?v\_DR))

(assert (Node (id (setID))

(UL ?v\_UL) (UC ?v\_UC) (UR ?v\_UR)

(ML ?v\_ML) (MC 0) (MR ?v\_MC)

(DL ?v\_DL) (DC ?v\_DC) (DR ?v\_DR)

(depth (+ ?v\_g 1)) (status 0) (parent ?v\_id) (f ?a2)

)

)

(if (= ?\*DEBUG\* 1) then (printout t "To uncover:" crlf

?v\_UL " " ?v\_UC " " ?v\_UR crlf

?v\_ML " 0 " ?v\_MC crlf

?v\_DL " " ?v\_DC " " ?v\_DR crlf

)

)

(bind ?a3 (getF (+ ?v\_g 1) ?v\_UL ?v\_UC ?v\_UR ?v\_ML ?v\_MC ?v\_DR ?v\_DL ?v\_DC 0))

(assert (Node (id (setID))

(UL ?v\_UL) (UC ?v\_UC) (UR ?v\_UR)

(ML ?v\_ML) (MC ?v\_MC) (MR ?v\_DR)

(DL ?v\_DL) (DC ?v\_DC) (DR 0)

(depth (+ ?v\_g 1)) (status 0) (parent ?v\_id) (f ?a3)

)

)

(if (= ?\*DEBUG\* 1) then (printout t "To uncover:" crlf

?v\_UL " " ?v\_UC " " ?v\_UR crlf

?v\_ML " " ?v\_MC " " ?v\_DR crlf

?v\_DL " " ?v\_DC " 0" crlf

)

)

(bind ?\*NODES\_CREATED\* (+ ?\*NODES\_CREATED\* 3))

(bind ?\*STEPS\_PASSED\* (+ ?\*STEPS\_PASSED\* 1))

)

(defrule create\_DownLeft

(declare (salience 100))

?f\_addr\_min <- (min ?min)

?f\_addr\_node <- (Node (id ?v\_id)

(UL ?v\_UL) (UC ?v\_UC) (UR ?v\_UR)

(ML ?v\_ML) (MC ?v\_MC) (MR ?v\_MR)

(DL 0) (DC ?v\_DC) (DR ?v\_DR)

(depth ?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 " Depth=" ?v\_g " F=" ?v\_f " Parent=" ?v\_parent crlf

(fact-slot-value ?f\_addr\_node UL) " " (fact-slot-value ?f\_addr\_node UC) " " (fact-slot-value ?f\_addr\_node UR) crlf

(fact-slot-value ?f\_addr\_node ML) " " (fact-slot-value ?f\_addr\_node MC) " " (fact-slot-value ?f\_addr\_node MR) crlf

(fact-slot-value ?f\_addr\_node DL) " " (fact-slot-value ?f\_addr\_node DC) " " (fact-slot-value ?f\_addr\_node DR) crlf

"====="crlf

)

)

(modify ?f\_addr\_node(status 1))

(bind ?a1 (getF (+ ?v\_g 1) ?v\_UL ?v\_UC ?v\_UR 0 ?v\_MC ?v\_MR ?v\_ML ?v\_DC ?v\_DR))

(assert (Node (id (setID))

(UL ?v\_UL) (UC ?v\_UC) (UR ?v\_UR)

(ML 0) (MC ?v\_MC) (MR ?v\_MR)

(DL ?v\_ML) (DC ?v\_DC) (DR ?v\_DR)

(depth (+ ?v\_g 1)) (status 0) (parent ?v\_id) (f ?a1)

)

)

(if (= ?\*DEBUG\* 1) then (printout t "To uncover:" crlf

?v\_UL " " ?v\_UC " " ?v\_UR crlf

"0 " ?v\_MC " " ?v\_MR crlf

?v\_ML " " ?v\_DC " " ?v\_DR crlf

)

)

(bind ?a2 (getF (+ ?v\_g 1) ?v\_UL ?v\_UC ?v\_UR ?v\_ML ?v\_MC ?v\_MR ?v\_DC 0 ?v\_DR))

(assert (Node (id (setID))

(UL ?v\_UL) (UC ?v\_UC) (UR ?v\_UR)

(ML ?v\_ML) (MC ?v\_MC) (MR ?v\_MR)

(DL ?v\_DC) (DC 0) (DR ?v\_DR)

(depth (+ ?v\_g 1)) (status 0) (parent ?v\_id) (f ?a2)

)

)

(if (= ?\*DEBUG\* 1) then (printout t "To uncover:" crlf

?v\_UL " " ?v\_UC " " ?v\_UR crlf

?v\_ML " " ?v\_MC " " ?v\_MR crlf

?v\_DC " 0 " ?v\_DR crlf

)

)

(bind ?\*NODES\_CREATED\* (+ ?\*NODES\_CREATED\* 2))

(bind ?\*STEPS\_PASSED\* (+ ?\*STEPS\_PASSED\* 1))

)

(defrule create\_DownCenter

(declare (salience 100))

?f\_addr\_min <- (min ?min)

?f\_addr\_node <- (Node (id ?v\_id)

(UL ?v\_UL) (UC ?v\_UC) (UR ?v\_UR)

(ML ?v\_ML) (MC ?v\_MC) (MR ?v\_MR)

(DL ?v\_DL) (DC 0) (DR ?v\_DR)

(depth ?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 " Depth=" ?v\_g " F=" ?v\_f " Parent=" ?v\_parent crlf

(fact-slot-value ?f\_addr\_node UL) " " (fact-slot-value ?f\_addr\_node UC) " " (fact-slot-value ?f\_addr\_node UR) crlf

(fact-slot-value ?f\_addr\_node ML) " " (fact-slot-value ?f\_addr\_node MC) " " (fact-slot-value ?f\_addr\_node MR) crlf

(fact-slot-value ?f\_addr\_node DL) " " (fact-slot-value ?f\_addr\_node DC) " " (fact-slot-value ?f\_addr\_node DR) crlf

"====="crlf

)

)

(modify ?f\_addr\_node(status 1))

(bind ?a1 (getF (+ ?v\_g 1) ?v\_UL ?v\_UC ?v\_UR ?v\_ML 0 ?v\_MR ?v\_DL ?v\_MC ?v\_DR))

(assert (Node (id (setID))

(UL ?v\_UL) (UC ?v\_UC) (UR ?v\_UR)

(ML ?v\_ML) (MC 0) (MR ?v\_MR)

(DL ?v\_DL) (DC ?v\_MC) (DR ?v\_DR)

(depth (+ ?v\_g 1)) (status 0) (parent ?v\_id) (f ?a1)

)

)

(if (= ?\*DEBUG\* 1) then (printout t "To uncover:" crlf

?v\_UL " " ?v\_UC " " ?v\_UR crlf

?v\_ML " 0 " ?v\_MR crlf

?v\_DL " " ?v\_MC " " ?v\_DR crlf

)

)

(bind ?a2 (getF (+ ?v\_g 1) ?v\_UL ?v\_UC ?v\_UR ?v\_ML ?v\_MC ?v\_MR 0 ?v\_DL ?v\_DR))

(assert (Node (id (setID))

(UL ?v\_UL) (UC ?v\_UC) (UR ?v\_UR)

(ML ?v\_ML) (MC ?v\_MC) (MR ?v\_MR)

(DL 0) (DC ?v\_DL) (DR ?v\_DR)

(depth (+ ?v\_g 1)) (status 0) (parent ?v\_id) (f ?a2)

)

)

(if (= ?\*DEBUG\* 1) then (printout t "To uncover:" crlf

?v\_UL " " ?v\_UC " " ?v\_UR crlf

?v\_ML " " ?v\_MC " " ?v\_MR crlf

"0 " ?v\_DL " " ?v\_DR crlf

)

)

(bind ?a3 (getF (+ ?v\_g 1) ?v\_UL ?v\_UC ?v\_UR ?v\_ML ?v\_MC ?v\_MR ?v\_DL ?v\_DR 0))

(assert (Node (id (setID))

(UL ?v\_UL) (UC ?v\_UC) (UR ?v\_UR)

(ML ?v\_ML) (MC ?v\_MC) (MR ?v\_MR)

(DL ?v\_DL) (DC ?v\_DR) (DR 0)

(depth (+ ?v\_g 1)) (status 0) (parent ?v\_id) (f ?a3)

)

)

(if (= ?\*DEBUG\* 1) then (printout t "To uncover:" crlf

?v\_UL " " ?v\_UC " " ?v\_UR crlf

?v\_ML " " ?v\_MC " " ?v\_MR crlf

?v\_DL " " ?v\_DR " 0" crlf

)

)

(bind ?\*NODES\_CREATED\* (+ ?\*NODES\_CREATED\* 3))

(bind ?\*STEPS\_PASSED\* (+ ?\*STEPS\_PASSED\* 1))

)

(defrule create\_DownRight

(declare (salience 100))

?f\_addr\_min <- (min ?min)

?f\_addr\_node <- (Node (id ?v\_id)

(UL ?v\_UL) (UC ?v\_UC) (UR ?v\_UR)

(ML ?v\_ML) (MC ?v\_MC) (MR ?v\_MR)

(DL ?v\_DL) (DC ?v\_DC) (DR 0)

(depth ?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 " Depth=" ?v\_g " F=" ?v\_f " Parent=" ?v\_parent crlf

(fact-slot-value ?f\_addr\_node UL) " " (fact-slot-value ?f\_addr\_node UC) " " (fact-slot-value ?f\_addr\_node UR) crlf

(fact-slot-value ?f\_addr\_node ML) " " (fact-slot-value ?f\_addr\_node MC) " " (fact-slot-value ?f\_addr\_node MR) crlf

(fact-slot-value ?f\_addr\_node DL) " " (fact-slot-value ?f\_addr\_node DC) " " (fact-slot-value ?f\_addr\_node DR) crlf

"====="crlf

)

)

(modify ?f\_addr\_node(status 1))

(bind ?a1 (getF (+ ?v\_g 1) ?v\_UL ?v\_UC ?v\_UR ?v\_ML ?v\_MC 0 ?v\_DL ?v\_DC ?v\_MR))

(assert (Node (id (setID))

(UL ?v\_UL) (UC ?v\_UC) (UR ?v\_UR)

(ML ?v\_ML) (MC ?v\_MC) (MR 0)

(DL ?v\_DL) (DC ?v\_DC) (DR ?v\_MR)

(depth (+ ?v\_g 1)) (status 0) (parent ?v\_id) (f ?a1)

)

)

(if (= ?\*DEBUG\* 1) then (printout t "To uncover:" crlf

?v\_UL " " ?v\_UC " " ?v\_UR crlf

?v\_ML " " ?v\_MC " 0" crlf

?v\_DL " " ?v\_DC " " ?v\_MR crlf

)

)

(bind ?a2 (getF (+ ?v\_g 1) ?v\_UL ?v\_UC ?v\_UR ?v\_ML ?v\_MC ?v\_MR ?v\_DL 0 ?v\_DC))

(assert (Node (id (setID))

(UL ?v\_UL) (UC ?v\_UC) (UR ?v\_UR)

(ML ?v\_ML) (MC ?v\_MC) (MR ?v\_MR)

(DL ?v\_DL) (DC 0) (DR ?v\_DC)

(depth (+ ?v\_g 1)) (status 0) (parent ?v\_id) (f ?a2)

)

)

(if (= ?\*DEBUG\* 1) then (printout t "To uncover:" crlf

?v\_UL " " ?v\_UC " " ?v\_UR crlf

?v\_ML " " ?v\_MC " " ?v\_MR crlf

?v\_DL " 0 " ?v\_DC crlf

)

)

(bind ?\*NODES\_CREATED\* (+ ?\*NODES\_CREATED\* 2))

(bind ?\*STEPS\_PASSED\* (+ ?\*STEPS\_PASSED\* 1))

)