Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное

учреждение высшего образования

ИРКУТСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ

ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Институт/Факультет | Институт информационных технологий и анализа | | | |
|  |  | наименование | |  |
| данных | |  | |  |
| Кафедра/Структурное подразделение | | | Автоматизированные системы | |
|  | | наименование (при наличии) | | |
| обработки информации и управления | | | | |

**ОТЧЕТ**

по лабораторной работе № \_2\_\_\_

по дисциплине \_\_Системы искусственного интеллекта\_\_\_\_

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Выполнил | АСУб-20-2 |  |  |  | Арбакова А.В. |
|  | шифр группы |  | подпись |  | Фамилия И.О. |
| Проверил |  |  |  |  | Столбов А.Б. |
|  | должность |  | подпись |  | Фамилия И.О. |

Иркутск – 20 \_23\_

**Лабораторная работа №1: Изучение основных возможностей и базовых команд среды CLIPS**

**Цель работы:** Изучение основных возможностей и базовых команд среды продукционного программирования CLIPS и освоение способов разработки экспертной системы.

**Задание:**

1. Изучение базовых команд и конструкций CLIPS
   1. Запустить систему CLIPS (файл clipswin.exe). Активизировать окно просмотра текущего списка фактов (подпункт «Facts Window» пункта «Windows» главного меню). Выполнить следующую последовательность действий, фиксируя после каждого шага состояние списка фактов:
      1. сбросить систему в исходное состояние командой (clear);
      2. выполнить начальную установку командой (reset) или комбинацией клавиш ^E;
      3. ввести 3 любых упорядоченных факта командой (assert), например: (assert (n n) (m m) (p p));
      4. повторно выполнить сброс командой (reset);
      5. установить 3 ранее вводимых упорядоченных факта в качестве исходных фактов, используя конструкцию (deffacts);
      6. выполнить сброс командой (reset).
   2. Активизировать дополнительно окно просмотра агенды (подпункт «Agenda Window» пункта «Windows» главного меню). Выполнить следующую последовательность действий, фиксируя после каждого шага состояния списка фактов и агенды:
      1. используя конструкцию (defrule), ввести три правила, такие, что антецеденты первых двух правил сопоставляются с комбинацией фактов, заданных ранее конструкцией (deffacts), а консеквенты этих правил добавляют новые факты, сопоставляемые с антецедентом третьего правила. Пусть, например, X, Y и Z – факты, заданные конструкцией (deffacts). Тогда структура вводимых правил может быть представлена следующим образом:

X & Y => V;

Y & Z => W;

V & W => U;

* + 1. выполнить по шагам активизацию правил (используя «горячую» комбинацию ^T).

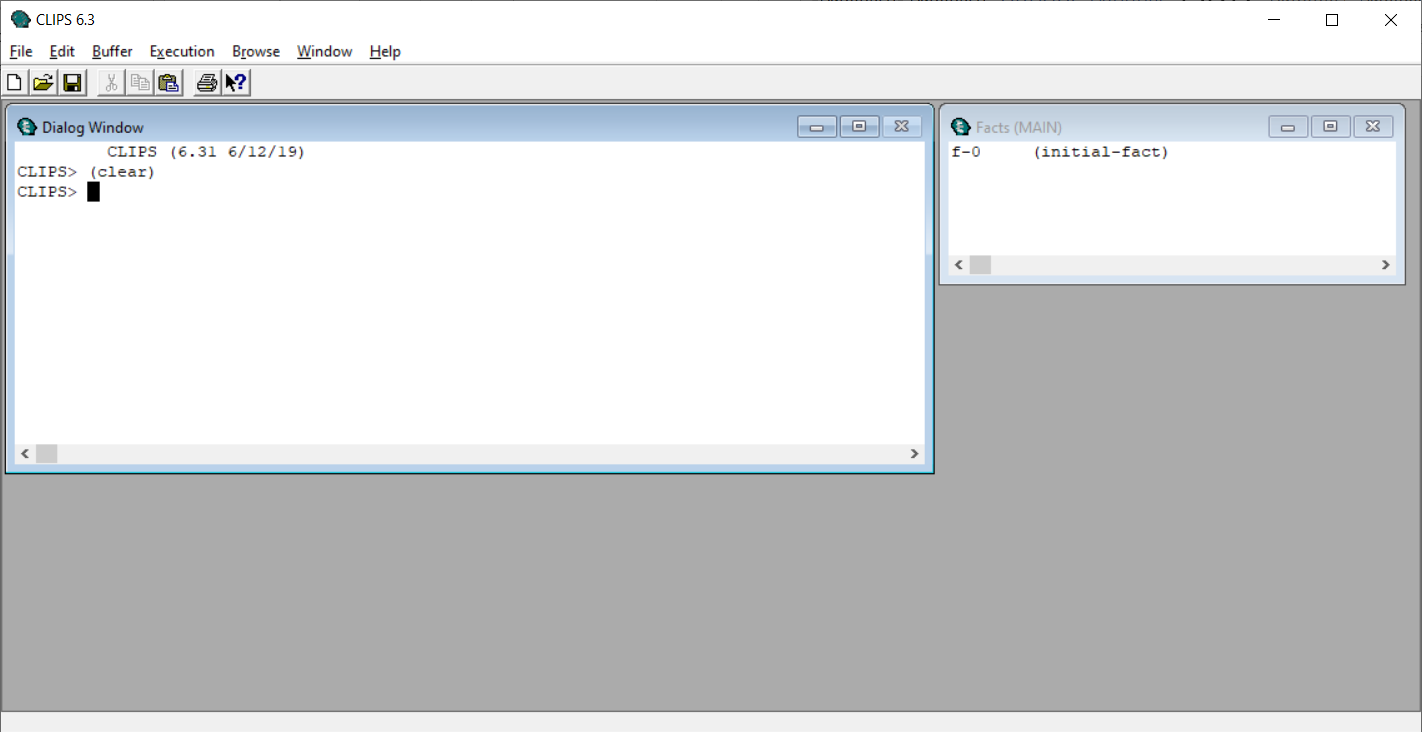
1. Разработка демонстрационной экспертной системы

Сформировать, пользуясь редактором clipsedt.exe, базу знаний демонстрационной ЭС и сохранить ее в файле rulebase.clp. ЭС должна вырабатывать рекомендации студенту накануне зачета и иметь четыре входные переменные («число дней до зачета», «количество несделанных лабораторных работ (в %)», «температура на улице» и «наличие осадков»), две промежуточные («свободное время» и «погода») и выходную переменную («рекомендуемые действия»). Диаграмма зависимости переменных показана на рисунке, в скобках указаны возможные имена переменных:

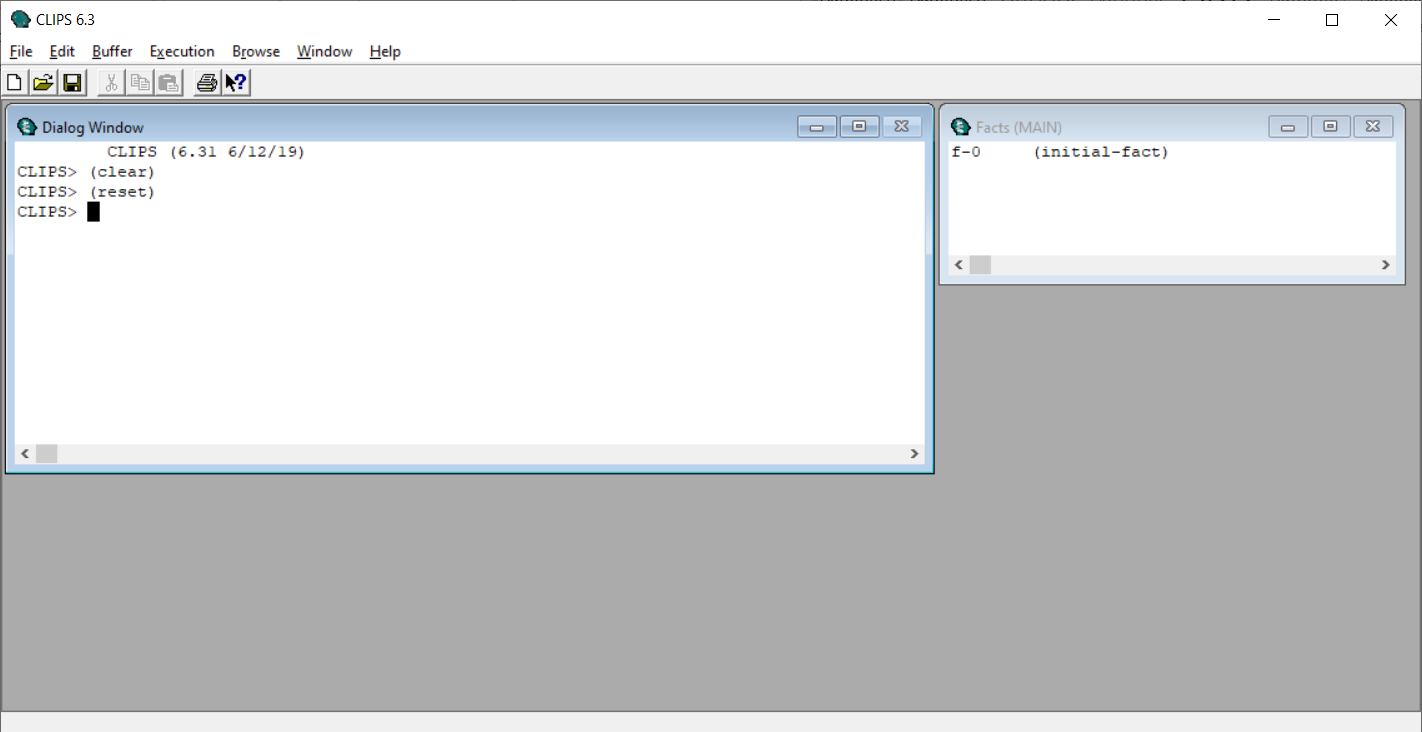
Общее количество правил в БЗ должно быть не менее 25. Количество значений переменных должно выбираться таким образом, чтобы БЗ отвечала требованию полноты, т.е. содержала правила, соответствующие любым сочетаниям значений переменных в левых частях правил. Например, если переменная «свободное время» имеет 3 значения («отсутствует», «мало» и «много»), а переменная «погода» – 2 значения («плохая» и «хорошая»), то максимальное число правил для определения переменной «действие» будет равно 6.

**Решение**:

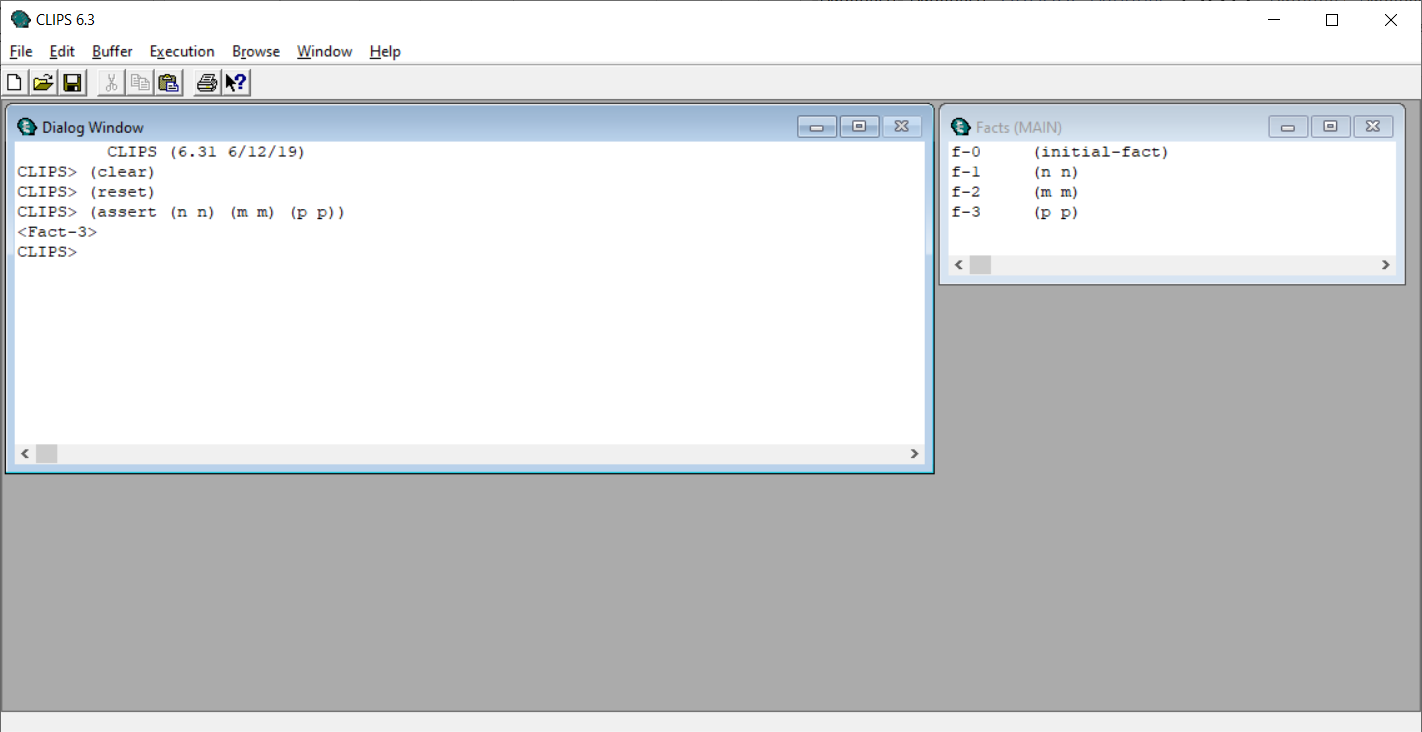
Сбросить систему в исходное состояние командой (clear):



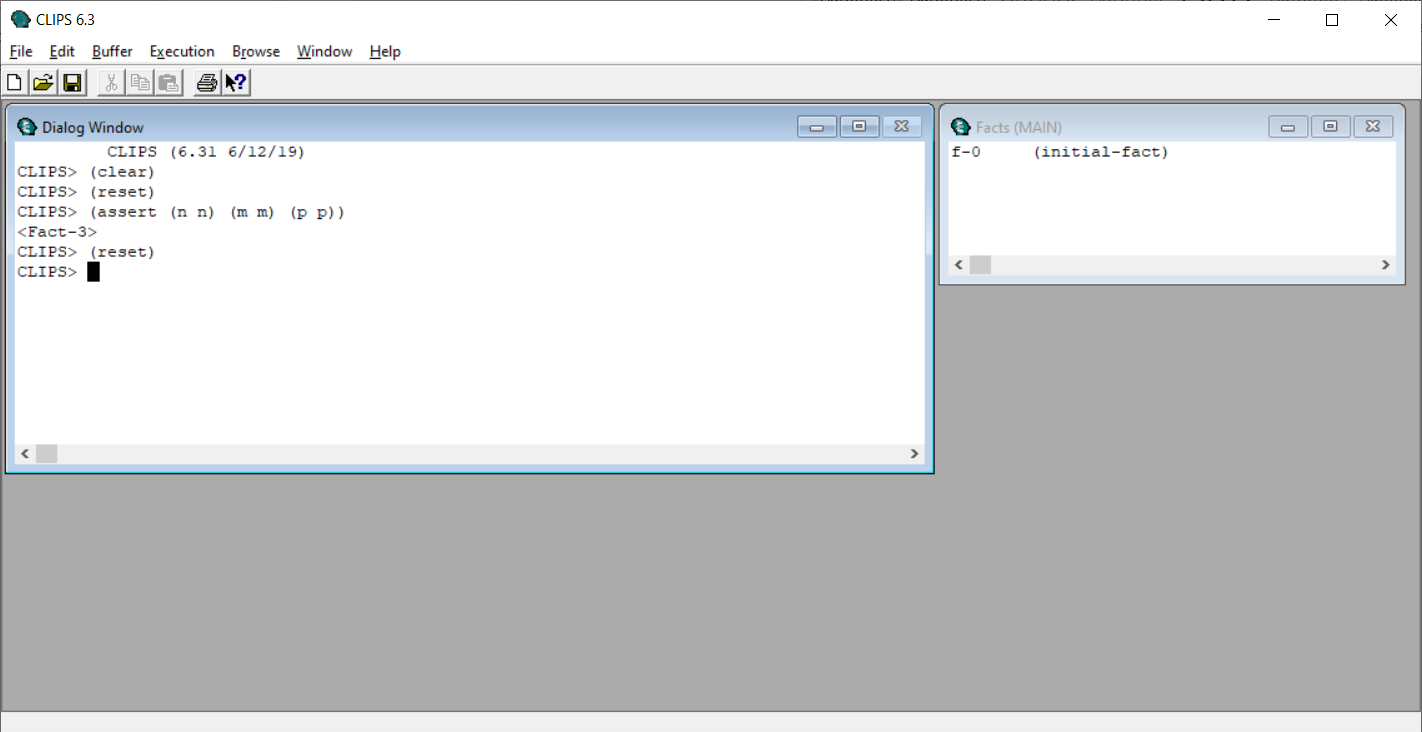
Выполнить начальную установку командой (reset) или комбинацией клавиш ^E:



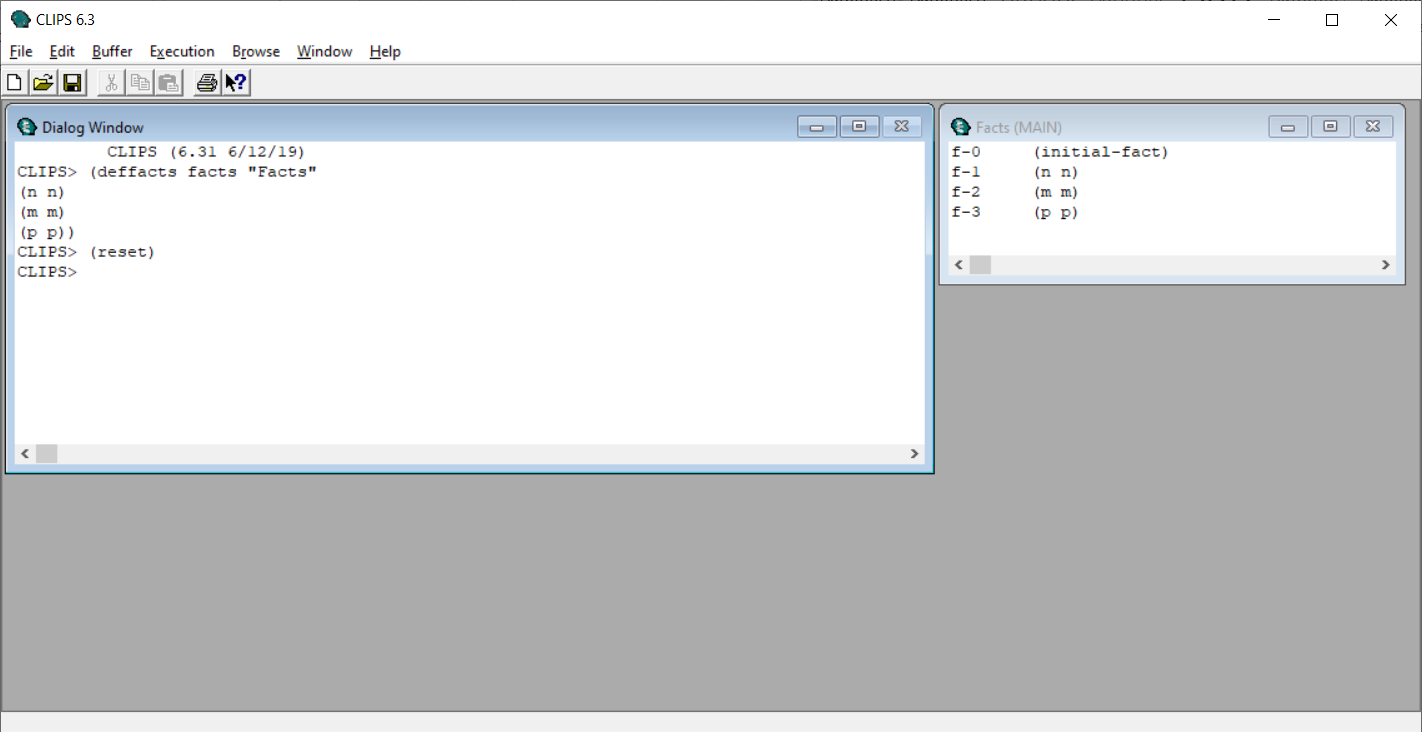
Ввести 3 любых упорядоченных факта командой (assert), например: (assert (n n) (m m) (p p)):



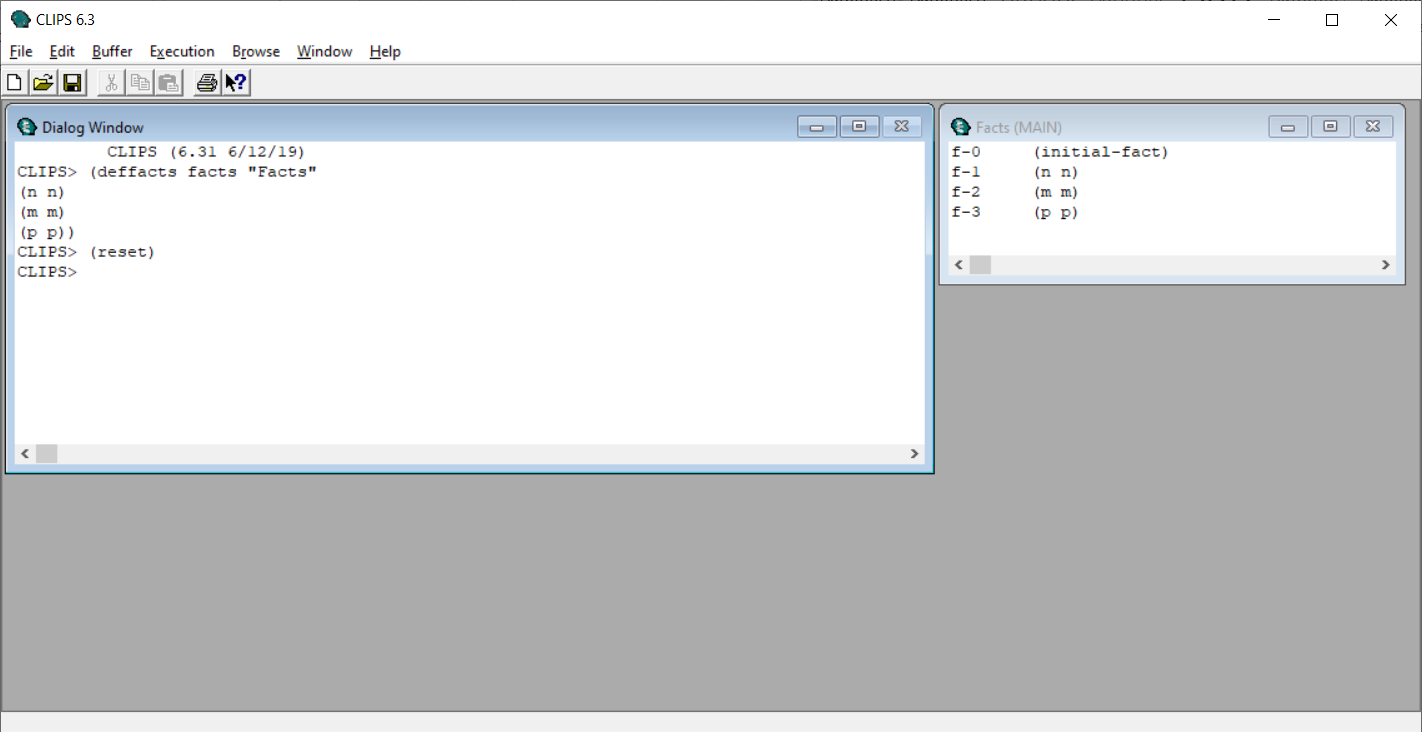
Повторно выполнить сброс командой (reset):



Установить 3 ранее вводимых упорядоченных факта в качестве исходных фактов, используя конструкцию (deffacts):



Выполнить сброс командой (reset):

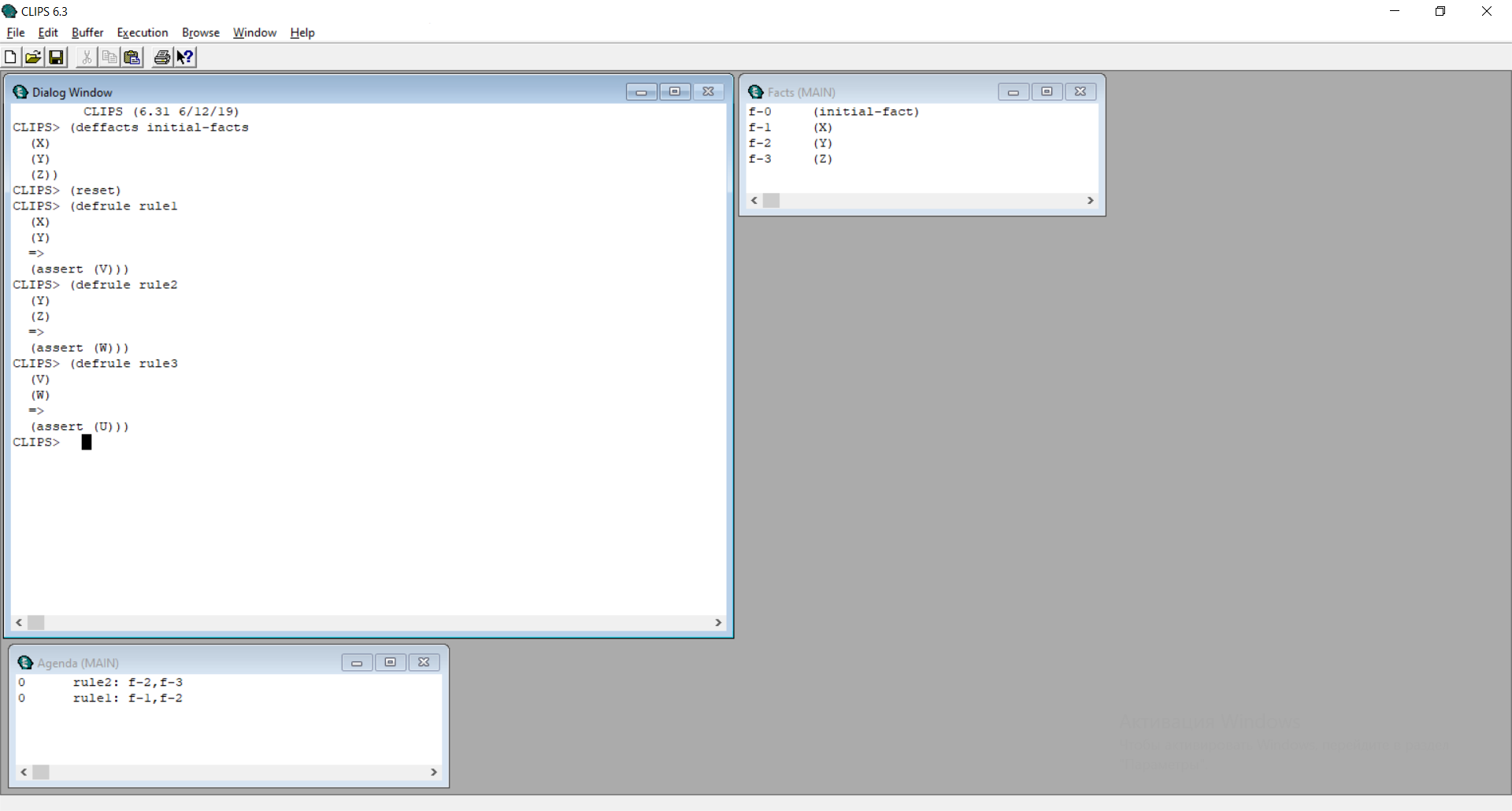


Используя конструкцию (defrule), ввести три правила, такие, что антецеденты первых двух правил сопоставляются с комбинацией фактов, заданных ранее конструкцией (deffacts), а консеквенты этих правил добавляют новые факты, сопоставляемые с антецедентом третьего правила. Пусть, например, X, Y и Z – факты, заданные конструкцией (deffacts). Тогда структура вводимых правил может быть представлена следующим образом:

X & Y => V;

Y & Z => W;

V & W => U;



Сформировать, пользуясь редактором clipsedt.exe, базу знаний демонстрационной ЭС и сохранить ее в файле rulebase.clp. ЭС должна вырабатывать рекомендации студенту накануне зачета и иметь четыре входные переменные («число дней до зачета», «количество несделанных лабораторных работ (в %)», «температура на улице» и «наличие осадков»), две промежуточные («свободное время» и «погода») и выходную переменную («рекомендуемые действия»).

Решение:

(defrule data-input

(initial-fact)

=>

(printout t crlf "Vvedite chislo dnei do zacheta (tseloe znachenie): ")

(bind ?days (read))

(assert (days ?days))

(printout t crlf "Vvedite chislo nesdelannyh laboratornyh rabot (v %) ")

(bind ?works (read))

(assert (works ?works))

(printout t crlf "Vvedite temperaturu na ulitse: ")

(bind ?temper (read))

(assert (temper ?temper))

(printout t crlf "Est' li na ulitse osadki? (da - 1/net - 0): ")

(bind ?rain (read))

(assert (rain ?rain))

(printout t crlf "Is there any white rabbit? (da - 1/net - 0) //HASN'T Realized: "))

(defrule R1

(days ?days)

(works ?works)

(test (= ?works 0))

=>

(printout t crlf crlf "Vse uzhe sdelano." crlf)

(assert (freetime "infinity"))

(assert (freetimecnst 0)))

(defrule R2

(days ?days)

(works ?works)

(test (and (and(> ?days 5) (<= ?days 7)) (and (<= ?works 50) (> ?works 0) )))

=>

(printout t crlf crlf "Svobodnogo vremeni mnogo" crlf)

(assert (freetime "mnogo"))

(assert (freetimecnst 1)))

(defrule R3

(days ?days)

(works ?works)

(test (and (and(> ?days 5) (<= ?days 7)) (and (<= ?works 100) (> ?works 50) )))

=>

(printout t crlf crlf "Svobodnogo vremeni ne ochen' mnogo" crlf)

(assert (freetime "ne\_ochen"))

(assert (freetimecnst 2)))

(defrule R4

(days ?days)

(works ?works)

(test (and (and(> ?days 3) (<= ?days 5)) (and (<= ?works 50) (> ?works 0) )))

=>

(printout t crlf crlf "Svobodnogo vremeni mnogo" crlf)

(assert (freetime "mnogo"))

(assert (freetimecnst 1)))

(defrule R5

(days ?days)

(works ?works)

(test (and (and(> ?days 3) (<= ?days 5)) (and (<= ?works 100) (> ?works 50) )))

=>

(printout t crlf crlf "Svobodnogo vremeni ne ochen' mnogo" crlf)

(assert (freetime "ne\_ochen"))

(assert (freetimecnst 2)))

(defrule R6

(days ?days)

(works ?works)

(test (and (= ?days 3) (and ( > ?works 0 ) (<= ?works 50) )))

=>

(printout t crlf crlf "Svobodnogo vremeni ne ochen' mnogo" crlf)

(assert (freetime "ne\_ochen"))

(assert (freetimecnst 2)))

(defrule R7

(days ?days)

(works ?works)

(test (and (= ?days 3) (and ( > ?works 50 ) (<= ?works 100) )))

=>

(printout t crlf crlf "Svobodnogo vremeni sovsem nemnogo. Pora delat'" crlf)

(assert (freetime "pora\_dalat"))

(assert (freetimecnst 3)))

(defrule R8

(days ?days)

(works ?works)

(test (and (= ?days 2) (and ( > ?works 0 ) (<= ?works 33) )))

=>

(printout t crlf crlf "Svobodnogo vremeni ne ochen' mnogo" crlf)

(assert (freetime "ne\_ochen"))

(assert (freetimecnst 2)))

(defrule R9

(days ?days)

(works ?works)

(test (and (= ?days 2) (and ( > ?works 33 ) (<= ?works 66) )))

=>

(printout t crlf crlf "Svobodnogo vremeni sovsem nemnogo. Pora delat'" crlf)

(assert (freetime "pora\_dalat"))

(assert (freetimecnst 3)))

(defrule R10

(days ?days)

(works ?works)

(test (and (= ?days 2) (and ( > ?works 66 ) (<= ?works 100) )))

=>

(printout t crlf crlf "Svobodnogo vremeni net -- ne uspevaem" crlf)

(assert (freetime "finish"))

(assert (freetimecnst 4)))

(defrule R11

(days ?days)

(works ?works)

(test (and (= ?days 1) (and ( > ?works 0 ) (<= ?works 25) )))

=>

(printout t crlf crlf "Svobodnogo vremeni ne ochen' mnogo" crlf)

(assert (freetime "ne\_ochen"))

(assert (freetimecnst 2)))

(defrule R12

(days ?days)

(works ?works)

(test (and (= ?days 1) (and ( > ?works 25 ) (<= ?works 50) )))

=>

(printout t crlf crlf "Svobodnogo vremeni sovsem nemnogo. Pora delat'" crlf)

(assert (freetime "pora\_dalat"))

(assert (freetimecnst 3)))

(defrule R13

(days ?days)

(works ?works)

(test (and (= ?days 1) (and ( > ?works 50 ) (<= ?works 100) )))

=>

(printout t crlf crlf "Svobodnogo vremeni net -- ne uspevaem" crlf)

(assert (freetime "finish"))

(assert (freetimecnst 4)))

(defrule R14

(days ?days)

(works ?works)

(test (and (= ?days 0) ( > ?works 0 )))

=>

(printout t crlf crlf "Nu kogda-to ono bylo. A seichas uzhe ne vazhno" crlf)

(assert (freetime "ppc"))

(assert (freetimecnst 5)))

(defrule R15

(temper ?temper)

(rain ?rain)

(test (> ?temper 25))

=>

(printout t crlf crlf "Pogoda ochen' horoshaya " crlf)

(assert (weather "v-good"))

(assert (weathercnst 1)))

(defrule R16

(temper ?temper)

(rain ?rain)

(test (and(and(>= ?temper 5)(< ?temper 25)) (= ?rain 0)) )

=>

(printout t crlf crlf "Pogoda horoshaya " crlf)

(assert (weather "good"))

(assert (weathercnst 2)))

(defrule R17

(temper ?temper)

(rain ?rain)

(test (and(and(>= ?temper 5)(< ?temper 25)) (<> ?rain 0)) )

=>

(printout t crlf crlf "Pogoda plohaya " crlf)

(assert (weather "bad"))

(assert (weathercnst 3)))

(defrule R18

(temper ?temper)

(rain ?rain)

(test (<= ?temper 5) )

=>

(printout t crlf crlf "Pogoda ochen' plohaya " crlf)

(assert (weather "v-bad "))

(assert (weathercnst 4)))

(defrule R19

(weathercnst ?weathercnst)

(freetimecnst ?freetimecnst)

(test (and(< ?freetimecnst 3)(= ?weathercnst 1)))

=>

(printout t crlf crlf "Mozhno idti gulyat'" crlf)

(assert (act "go")))

(defrule R20

(weathercnst ?weathercnst)

(freetimecnst ?freetimecnst)

(test (= ?freetimecnst 5))

=>

(printout t crlf crlf "Po povodu pogodi ne znayu, no uchit' uje pozdno" crlf)

(assert (act "nth")))

(defrule R21

(weathercnst ?weathercnst)

(freetimecnst ?freetimecnst)

(test (= ?freetimecnst 0))

=>

(printout t crlf crlf "Po povodu pogodi ne znayu -- gotovsya k sleduyuchey sessii..." crlf)

(assert (act "botan")))

(defrule R22

(weathercnst ?weathercnst)

(freetimecnst ?freetimecnst)

(test (and(= ?freetimecnst 4)(<> ?weathercnst 5)))

=>

(printout t crlf crlf "Nado uchit'!" crlf)

(assert (act "learn")))

(defrule R23

(weathercnst ?weathercnst)

(freetimecnst ?freetimecnst)

(test (and(= ?freetimecnst 3)(= ?weathercnst 2)))

=>

(printout t crlf crlf "Luchshe uchit'sya" crlf)

(assert (act "learn")))

(defrule R24

(weathercnst ?weathercnst)

(freetimecnst ?freetimecnst)

(test (and(= ?freetimecnst 2)(= ?weathercnst 2)))

=>

(printout t crlf crlf "As u wish" crlf)

(assert (act "auw")))

(defrule R25

(weathercnst ?weathercnst)

(freetimecnst ?freetimecnst)

(test (and(= ?freetimecnst 1)(= ?weathercnst 2)))

=>

(printout t crlf crlf "As u wish" crlf)

(assert (act "auw")))

(defrule R26

(weathercnst ?weathercnst)

(freetimecnst ?freetimecnst)

(test (and( or (= ?freetimecnst 2)( = ?freetimecnst 1))(= ?weathercnst 3)))

=>

(printout t crlf crlf "Luchshe uchit'" crlf)

(assert (act "glearn")))

(defrule R27

(weathercnst ?weathercnst)

(freetimecnst ?freetimecnst)

(test (and(= ?freetimecnst 3)(= ?weathercnst 3)))

=>

(printout t crlf crlf "Luchshe uchit'" crlf)

(assert (act "glearn")))

(defrule R28

(weathercnst ?weathercnst)

(freetimecnst ?freetimecnst)

(test (and(> ?freetimecnst 0) (= ?weathercnst 4)))

=>

(printout t crlf crlf "Luchshe uchit'" crlf)

(assert (act "glearn")))

Пример выполнения:

