```
входные данные: i;                                          // по ссылке
начало
    если i = 0, то
        i ← 1 ;
        вернуть „ложь“ ;
    иначе                                                   // i ≠ 0
        вернуть „истина“ ;
```

Рис. 1: алгоритм атомарной проверки и установки

```
входные данные: i;                                          // по ссылке
начало
    t ← 1 ;
    обмен(i, t);                                            // атомарная
    вернуть (t ≠ 0) ;
```

Рис. 2: связь атомарной проверки и установки и обмена

```
функция Simple_Reflex_Vacuum_Agent ([положение, состояние]):
    входные данные: положение: ∈ {A, B};
    состояние: ∈ {Чисто, Грязно};
    начало
        если состояние = Грязно то
            вернуть Всасывать;
        иначе если положение = A то
            вернуть Вправо;
        иначе если положение = B то
            вернуть Влево;
```

Рис. 3: функция простого рефлексивного агента-пылесоса

```
функция Simple_ Problem_ Solving_ Agent (восприятие):
    входные данные: восприятие: от датчиков;
    вспомогательные данные: действия: последовательность
                                действий, первоначально пустая;
    состояние: ∃ описание текущего состояния мира;
    цель: первоначально неопределённая;
    задача: формулировка задачи;
    начало
        состояние ← Обновить_Состояние(состояние, восприятие);
        если действия = ∅ то
            цель ← Сформулировать_Цель(состояние);
            задача ← Сформулировать_Задачу(состояние, цель);
            действия ← Поиск(задача);
        действие ← Голова(действия);
        действия ← Хвост(действия);
        вернуть действие;
```

Рис. 4: функция простого агента, решающего задачу

```
функция Depth_Limited_Tree_Search (задача, ℓ):
    входные данные: ℓ: предел глубины поиска;
    результат: решение или «неудача» или «отсечение»;
    начало
        вернуть
        Recursive_DLS(Make_Node(задача.начальное_состояние),
        задача, ℓ);
функция Recursive_DLS(узел, задача, ℓ):
    начало
        было_отсечение ← «ложь»;
        если Проверка_Цели(задача, узел.состояние), то
            вернуть узел.решение;
        иначе если узел.глубина = ℓ, то
            вернуть «отсечение»;
        иначе
            для каждого преемник в Развернуть(узел, задача)
            выполнять
                результат ← Recursive_DLS(преемник, задача, ℓ);
                если результат = «отсечение», то
                    было_отсечение ← «истина»;
                иначе если результат ≠ «неудача», то
                    вернуть результат;
        если было_отсечение, то
            вернуть «отсечение»;
        иначе
            вернуть «неудача»;
```

Рис. 5: рекурсивная реализация поиска с ограничением глубины