УДК 002.53:004.89

МОДЕЛИРОВАНИЕ РОБОТА, УПРАВЛЯЕМОГО РЕЧЕВЫМИ СИГНАЛАМИ

Ю.А. Загорулько

Институт систем информатики им. А.П. Ершова СО РАН, г. Новосибирск
E-mail: zagor@iis.nsk.su

Представлен подход к разработке симулятора робота, управляемого речевыми сигналами. Симулятор реализован средствами программной среды Semp-TAO, поддерживающей гибкую интегрированную технологию создания интеллектуальных систем, он имеет расширяемую систему команд и легко настраивается на предметную область.

Ключевые слова:
Система речевого управления роботом, симулятор робота, язык управления роботом, операторная схема, программная среда, система правил-продукций.

Key words:
A speech control system for robot, robot simulator, control language, operator scheme, program environment, production rule system.

Введение

Интеллектуальные системы управления сложными техническими устройствами, в том числе роботами [1, 2], получили широкое распространение. В связи с тем, что стали доступными промышленные распознаватели и синтезаторы речи, такое управление все чаще осуществляется на естественном языке [3, 4].

Разработка системы речевого управления роботом является сложным и трудоемким процессом. Сначала необходимо распознать речевое сообщение, а затем его текст перевести в последовательность команд языка управления роботом и обеспечить их исполнение. Единственным претендующим на полноту критерием правильности понимания команды роботом является выполнение им требуемых действий. Проводить весь процесс отладки системы над «живым» роботом было бы дорого и неразумно. Поэтому при создании системы речевого управления роботом для упрощения процесса разработки и его ускорения, а также повышения качества всех компонентов системы имеет смысл использовать не робота, овеществленного в «металле», а его программную модель или симулятор. Основным требованием к такому симулятору является точное выполнение команд, подаваемых роботу. Оно может состоять, например, в изменении положения объектов на экране монитора или выдаче речевых сообщений, как бы исходящих от самого робота.

Использование симулятора позволяет лучше и точнее подобрать и отладить систему команд управления роботом. Заказчик уже на начальных этапах работы может судить по модели робота о его возможностях и вносить желаемые коррективы. Кроме того, использование симулятора робота позволяет более тонко настроить систему речевого управления, облегчает и ускоряет разработку и отладку лингвистического процессора.

В данной статье рассматривается подход к разработке симулятора робота, выполненной в рамках исследований по созданию системы речевого управления интеллектуальным роботом, которые велись Российским НИИ искусственного интеллекта и Институтом систем информатики СО РАН совместно с германским Институтом прикладных систем обработки знаний (FAW, Ulm). Интеллектуальный робот действует в помещении, состоящем из нескольких комнат, и может выполнять устные команды типа *«Перейди в комнату номер 5»* или *«Отнеси компьютер в третью комнату»*, а также отвечать на такие вопросы, как *«Где ты находишься?»*, *«Что расположено в этой комнате?»* и т. п. Система управления интеллектуальным роботом включает: подсистему речевого ввода, лингвистический процессор, транслирующий команду, выраженную на естественном языке, в формальное представление, а также синтезатор речи, озвучивающий сообщения робота.

Система речевого управления интеллектуальным роботом (рис. 1), обеспечивает полный цикл исполнения команды, подаваемой роботу.

Команда, произнесенная оператором, попадает на вход распознавателя речи, который преобразует ее в текст на естественном языке. Затем этот текст переводится лингвистическим процессором в последовательность команд. Команды, представленные на формальном языке, подаются на вход подсистемы управления роботом. Эта подсистема выполняет требуемые действия, что может привести к изменению состояния мира робота. Все изменения отображаются на экране визуализатором состояний, что позволяет наглядно демонстрировать результат выполнения команды. Кроме того, если поданная команда требует ответа, то подсистемой исполнения команд будет сгенерирован соответствующий текст ответа на естественном языке. Этот ответ преобразуется синтезатором речи в звуковой и сообщается пользователю. Система действует по замкнутому циклу, т. е. после исполнения роботом одной команды и получения реакции на нее оператор может подать следующую команду.

Рис. 1. *Функциональная схема системы речевого управления роботом*

1. Характеристика «способностей» робота

Мир моделируемого робота состоит из нескольких комнат, в которых могут размещаться объекты. Выделяются такие классы объектов, как мебель и оборудование.

В модели мира робот рассматривается и как объект, и как субъект. Как объект робот характеризуется такими же свойствами, что и оборудование. Как субъект, он должен уметь переносить мебель и оборудование из одной комнаты в другую и отвечать на вопросы пользователя. Поэтому основными функциями робота являются: *найти, взять, положить, переместить, перейти* и т. п.

Этим функциям соответствует набор операторов. Выделяется два уровня операторов. Первый уровень образуют операторы, доступные пользователю. Именно их он использует для формирования задания роботу. Второй уровень составляют операторы, реализующие операторы первого уровня.

Операторами первого уровня являются: **Принеси, Отнеси, Переместить, Перейди, Где_находится** и **Что_находится**.

Второй уровень образуют следующие операторы: **Найди**, **Возьми**, **Освободи**, **Поставь**, **Сообщи**.

Для каждого оператора имеется своя операторная схема, которая определяет условия, порядок и результаты выполнения оператора. В отличие от систем STRIPS и ABSTRIPS [5], использующих линейные операторные схемы, в нашей системе применяются рекурсивные схемы. Например, оператору второго уровня **Найди** соответствует следующая схема:

```
Найди (это){
    если местоположения это и робота совпадают то
        запомнить текущее местоположение робота в $место;
        выдай как результат ($место);
    иначе
        пометить текущую комнату как уже осмотренную;
        если соседняя комната еще не осмотрена то
            запомнить местоположение соседней комнаты в $новое_место;
            Перейди ($новое_место);
            Найди (это);
        иначе
            Сообщи (это, "не найдено");
}
```

Приведем также схему одного оператора первого уровня **Принеси**:

```
Принеси (это){
    запомнить текущее местоположение робота в $сюда;
    $место:=Найди (это);
    Переместить (это, $место, $сюда);
}
```

2. Формальный язык управления роботом

Для непосредственного управления роботом разработан формальный язык FOROL (FOrmal RObot Language), реализующий описанные выше операторные схемы. Именно в операторы этого языка отображаются команды пользователя. Язык FO-

ROL включает операторы *WhereIs*, *GO* и *MOVE*, аргументами которых являются объекты и комнаты. Рассмотрим синтаксис и семантику этих операторов.

Оператор *WhereIs* обеспечивает ответы на вопросы пользователя о месте нахождения робота и объектов. Он имеет следующий вид:

WhereIs (what: object, where: room),

здесь и в других операторах *object* – описание объекта, а *room* – описание комнаты.

В операторе *WhereIs* один или оба аргумента могут быть неопределенными, поэтому его семантика зависит от того, какие аргументы заданы, а какие нет.

Так, если задан только первый аргумент, то результатом выполнения оператора *WhereIs* является поиск объекта *object* с указанными характеристиками и выдача сообщения о том, где он находится. В случае неудачи выдается соответствующее сообщение.

Если задан только второй аргумент, то выдаются характеристики всех объектов, находящихся в комнате *room*. В случае, если не заданы оба параметра, сообщается обо всех объектах, находящихся в той же комнате, что и робот.

Оператор *GO* имеет всего один аргумент:

GO (to: room).

Этот оператор перемещает робота в комнату *room*. Если это невозможно, выдается соответствующее сообщение.

В операторе *MOVE* отображаются все команды пользователя, требующие перемещения объектов. Он имеет следующий синтаксис:

MOVE (what: object, from: room1, to: room2).

В операторе *MOVE* обязательным является только первый аргумент. Семантика этого оператора, как и *WhereIs*, зависит от того, какие аргументы заданы.

Так, если заданы все три аргумента, предмет *object* перемещается из комнаты *room1* в комнату *room2*. Если заданы только два аргумента *what* и *from*, предмет *object* перемещается из комнаты *room1* в комнату, в которой находится в данный момент робот. Когда заданы только аргументы *what* и *to*, предмет *object* должен быть найден и перемещен в комнату *room2*. Если задан только аргумент *what*, предмет *object* должен быть найден и перемещен в комнату, в которой находится робот.

Во всех вариантах оператора *MOVE* в случае неудачи выдается соответствующее сообщение. Например: *«Зеленый стул не найден»*, *«Нет комнаты с номером 20»*, *«Компьютер уже находится в комнате 5!»* и т. д.

3. Реализация симулятора робота

Симулятор робота реализован с помощью интегрированной программной среды Semp-TAO [6, 7], поддерживающей технологию создания интеллектуальных систем, включающую средства для описания сложных по структуре и семантике предметных областей и позволяющую сочетать логический вывод и вычисления над неточно заданными значениями. Все понятия предметной области, правила вывода и визуализатор состояний специфицированы на объектно-ориентированном языке представления и обработки знаний этой среды.

3.1. Представление мира робота

Поскольку объекты мира робота и он сам занимают определенное место в пространстве, то все они представляются в виде геометрических фигур или имеют специальный атрибут, значением которого является некоторый геометрический объект, отражающий пространственные характеристики данного объекта мира робота. Для описания всех геометрических объектов вводится общее понятие фигура, которому в языке Semp-TAO соответствует класс *SHAPE*:

```
class SHAPE
        x, y: integer;
        left, right, top, bottom: integer;
end;
```

Здесь *x, y* – координаты центра фигуры, а *left, right, top, bottom* – координаты ее границ.

На основе класса *SHAPE* строятся классы *CIRCLE* и *RECTANGLE*, представляющие, соответственно, окружность и прямоугольник.

Для описания комнаты введен класс ROOM, который является наследником класса *RECTANGLE*. Для представления всех объектов мира вводится абстрактный класс *OBJECT*, который включает общие для всех объектов характеристики, такие как имя, цвет, и размер. Для задания геометрических характеристик объекта и указания его местоположения в пространстве вводится специальный параметр *loc*, значением которого является прямоугольник, в который вписан объект. Кроме того, с каждым объектом связывается его графический образ, в виде которого он будет отображаться на экране.

Кроме объектов вводятся различные отношения между ними. Например, отношение *INSIDE*, указывающее, что некоторый объект находится в другом. Заметим, что это декларативное отношение имеет и вычислительную интерпретацию, т. е. включает набор ограничений (*constraints*), связывающих пространственные характеристики двух объектов:

```
relation INSIDE (what: SHAPE; into: SHAPE)
constraints
        what.loc.left >= into.loc.left;
        what.loc.right <= into.loc.right;
        what.loc.top <= into.loc.top;
        what.loc.bottom >= into.loc.bottom;
end;
```

3.2. Симуляция действий робота

Симуляции действий робота выполняется с помощью системы правил-продукций, реализующих операторные схемы робота. Эта система обрабатывает команды, поступающие на вход симулятора, редактирует и отображает на экране монитора со-

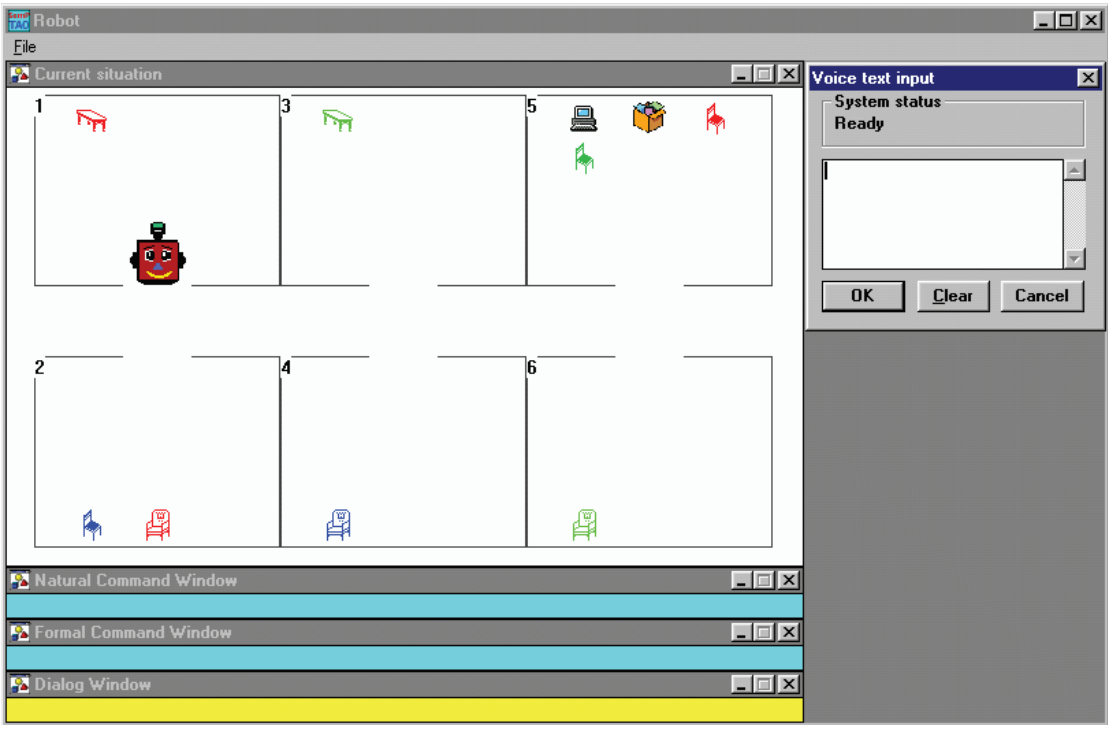

Рис. 2. *Пользовательский интерфейс симулятора*

ответствующие сцены (рис. 2), а также синтезирует и выдает необходимые сообщения.

Все множество правил системы разбито на шесть групп, каждая из которых отвечает за определенный вид или этап обработки задания. Первая группа правил (*$Do*) распознает тип команды и подготавливает ее исполнение. Каждая из остальных групп правил соответствует одной или нескольким близким по семантике операторным схемам. Так, вторая группа правил (*$Where*) обеспечивает выдачу ответов на вопросы типа *«Где находится компьютер?»*. Третья группа (*$Go*) выполняет перемещение робота, четвертая (*$Move*) – перемещение объекта. Пятая группа (*$Find*) осуществляет поиск нужного объекта. Шестая группа (*$Draw*) размещает объекты внутри комнаты и рисует плансхему на экране. Благодаря встроенному в программную среду Semp-TAO потоковому механизму удовлетворения ограничений [8], такое размещение выполняется автоматически.

Во время исполнения некоторых групп правил могут вызываться другие группы. Имеющиеся в языке среды гибкие средства управления активацией правил позволяют делать это естественным образом. Так, группа правил *$Where* во время своей работы всегда вызывает группу правил *$Find* для нахождения необходимого объекта. Последняя, в свою очередь, может вызвать группу *$Go*. При работе группы правил *$Move* также могут инициализироваться вызовы групп правил *$Find* и *$Go*.

Чтобы дать представление о том, как работает система правил-продукций, рассмотрим несколько правил из группы *$Find*. Первое из них имеет вид:

```
rule GoIni
exist
        $find: FIND (what: OBJECT (name: $name,
        color: $color), where:?),
        $last: LAST (where: $room1),
        NEXT (from: $room1, to: $room2)
not
        WAS (where: $room2)
=>
        delete $last;
        new GO (to: $room2);
        call $Go;
end;
```

Это правило в случае неудачного поиска в текущей комнате (*$room1*) инициализирует перемещение робота в следующую комнату (*$room2*), в которой он еще не был (на это указывает подобразец в условии правила *not WAS (where: $room2)*). Это осуществляется путем генерации команды *GO (to:$room2)* и вызовом соответствующей группы правил *(call $Go)*.

Второе правило:

```
rule NotFound
exist
        $find: FIND (what: OBJECT (name: $name,
        color: $color), where:?),
        $last: LAST (where: $room1),
        NEXT (from: $room1, to: $room2),
        WAS (where: $room2)
=>
        Say («The « + $color + « « + $name + « is not
        found!»);
        delete $last;
end;
```

соответствует ситуации, когда искомый объект не обнаружен ни в одной из комнат. (В комнате *$room1* объекта нет, а в *$room2* и во всех остальных комнатах робот уже был, о чем говорит наличие в сети объекта *WAS (where: $room2))*. Сообщение о том, что объект не обнаружен, озвучивается синтезатором речи, который вызывается с помощью функции *Say*.

Заключение

Представлен подход к разработке симулятора робота, управляемого речевыми сигналами. Благодаря тому, что симулятор робота реализован средствами объектно-ориентированной программной среды Semp-TAO, поддерживающей гибкую интегрированную технологию создания интеллектуальных систем, он имеет расширяемую систему команд и легко настраивается на предметную область.

Рассмотренный в статье программный симулятор робота прошел тестирование в составе системы речевого управления роботом и показал устойчивую работоспособность на большом объеме тестов. Его использование позволило более точно подобрать и опробовать систему команд управления роботом и значительно ускорило разработку и отладку лингвистического процессора.

СПИСОК ЛИТЕРАТУРЫ

1. Литвинцева Л.В., Ульянов С.В., Танака Т., Охви Д., Ямафудзи К. Интеллектуальное управление мобильным роботом сервисного обслуживания // Программные продукты и системы. – 1996. – № 3. – С. 35–43.
2. Nakhaeinia D., Tang S.H., Mohd Noor S.B., Motlagh O. A review of control architectures for autonomous navigation of mobile robots // International Journal of the Physical Sciences. – 2011. – V. 6 (2). – P. 169–174.
3. Luth T.C., Langle Th., Herzog G., Stopp E., Rembold V. KANTRA: Human-Machine Interaction for Intelligent Robots using Natural Language // 3rd IEEE Intern. Workshop on Robot and Human Communication, ROMAN'94. – Nagoya, Japan, 1994. – P. 106–111.
4. Bartneck C., Okada M. Robotic User Interfaces // Proc. of the Human and Computer Conference (HC2001). – Aizu-Wakamatsu, Japan, 2001. – P. 130–140.
5. Ефимов Е.И. Решатели интеллектуальных задач. – М.: Наука, 1982. – 320 с.
6. Загорулько Ю.А., Попов И.Г., Щипунов В.В. Интегрированная технологическая среда для создания систем обработки знаний // Известия РАН. Сер. Теория и системы управления. – 1995. – № 5. – С. 210–213.
7. Загорулько Ю.А., Попов И.Г. Представление знаний в интегрированной технологической среде Semp-TAO // Проблемы представления и обработки не полностью определенных знаний / под ред. И.Е. Швецова. – М.-Новосибирск: Рос. НИИ искусственного интеллекта, 1996. – С. 59–74.
8. Нариньяни А.С. Недоопределенность в системах представления и обработки знаний // Известия АН СССР. Сер. Техническая кибернетика. – 1986. – № 5. – С. 3–28.

Поступила 21.09.2011 г.