МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ имени М.В. ЛОМОНОСОВА
ФАКУЛЬТЕТ ВЫЧИСЛИТЕЛЬНОЙ МАТЕМАТИКИ И КИБЕРНЕТИКИ
КАФЕДРА СИСТЕМНОГО ПРОГРАММИРОВАНИЯ

Введение в дипломную работу

Генерация моделей на унифицированном языке моделирования по описаниям предметных областей и задач планирования

Исполнитель:
студент 527 группы Елисеев Владислав

Научный руководитель:
к.ф.-м.н. Малышко Виктор Васильевич

Москва
2013

Одна из задач в области искусственного интеллекта - задача планирования. Планированием называется процесс выработки последовательности действий, позволяющей достичь цели [1]. Задача планирования часто встает у агентов, которым нужно найти последовательность действий, выполнив которые он достигнет некоторой своей цели. Агентом можно назвать все, что воспринимает свою среду с помощью некоторых датчиков и воздействует на нее с помощью некоторых механизмов. Примерами использования планирования может служить следующее: планирование управлением механическими приводами робота-мусоросборника, планирование распределения транспортных средств для перевозок различных видов топлива и сырья на нефтеперерабатывающем заводе.

Контекст задачи планирования, модель мира, в котором возникает задача, – называется предметной областью. Они могут сильно отличаться друг от друга. Они могут быть детерминированными или стохастическими, статическими или динамическими, полностью наблюдаемыми или частично наблюдаемыми, и т.д. Многообразие моделей влияет на многообразие и сложность соответствующих алгоритмов решения задач планирования, на сложность и архитектуру агентов, на решение задачи планирования и её представление.

В работе рассматриваются полностью наблюдаемые, детерминированные, конечные, статические и дискретные модели мира, называемые иначе классическими моделями, или средами [1]. Для агентов, действующих в классических средах, характерно выделение трех основных понятий: *состояние среды* (мира), *действие* (механизмы воздействия на среду), *цель* (целевое состояние мира), и встает необходимость определить, как представляется информация об этих понятиях.

В 1971 году был разработан формальный текстовый язык STRIPS [2] – STanford Research Institute Problem Solver – для представления задач планирования в классических средах, что послужило толчком к развитию классических алгоритмов планирования и планировщиков. В этом языке состояния задаются в виде некоторого набора фактов об объектах и отношениях между ними, которые считаются истинными в этом состоянии. Действия задаются с использованием набора ограничений на состояние (предусловие), и набора положительных и отрицательных фактов (эффекты). *Предусловие* должно выполняться для того, чтобы применение действия было допустимым в дан-

ном состоянии. *Эффект* задает то, как меняется состояние при применении действия – положительные факты добавляются к состоянию, отрицательные – удаляются. *Цель*, или целевое состояние, задается как набор ограничений, которые должны быть удовлетворены в данном состоянии. Также в STRIPS вводится гипотеза замкнутости мира (CWA, *Closed World Assumption*), которая означает, что факты, не перечисленные в описании состояния, считаются ложными.

В 1987 году был предложен язык ADL – Action Description Language *(язык описания действий)* – похожий на STRIPS, но включающий в себя несколько особенностей, из которых можно отметить следующие:

- предположение об открытом мире - факты, не перечисленные в состоянии, считаются неизвестными;
- возможность использования кванторов и дизъюнктов для задания цели;
- возможность использования условных эффектов;
- наличие встроенного предиката равенства для сравнения объектов;
- типизация переменных;
- и т.д.

Помимо языков STRIPS и ADL, вдохновленные ими, развивались и другие текстовые языки представления знаний, каждый из которых использовал свой синтаксис, семантику и другие возможности. В 1988 году появился язык PDDL [3] – Planning Domain Definition Language *(язык описания предметных областей и задач планирования)* – как попытка стандартизации существующих на тот момент языков описания предметных областей и задач планирования. Также это сделало возможным создание IPC – International Planing Competition – международных соревнований по созданию планировщиков.

Рассмотрим примеры PDDL-описаний для игры «Sokoban» *(пер. кладовщик)*, предложенные на IPC в 2008 году. Мир в данной игре представляет собой некоторую область, поделенную на клетки одинакового размера, которые бывают проходимыми и непроходимыми (фактически, это лабиринт).

В некоторых клетках расположены предметы – это могут быть камни, сундуки, ключи и т.п. Другие клетки имеют пометки целевые – углубления в полу для камней, замочные скважины для ключей (для люков в полу) и т.п. Количество целевых клеток и предметов совпадает. В игре есть игрок, который передвигается по клеткам лабиринта. В одной клетке не могут находиться игрок и предмет одновременно. Целью игрока является расположить все предметы по их целевым клеткам, причем не важно, в каком порядке и каким образом они будут расположены в целевых клетках. За один ход игрок может передвинуть предмет, находящийся в соседней клетке по направлению движения, в клетку за ней, если она свободна и проходима. Фрагмент описания предметной области игры «Sokoban» представлен далее:

```
(define (domain sokoban-sequential)
  ;; определяем предметную область

  (:requirements :typing :action-costs)
  ;; указываем какие возможности PDDL используются
  ;; :typing -- используется типизация переменных
  ;; :action-costs -- действия имеют стоимость

  (:types thing location direction - object
          player stone - thing)
  ;; описываем иерархию типов: thing, location, direction --
  ;; наследуют базовый тип object
  ;; player, stone -- наследуют уже определенный тип thing

  ;; далее описываются предикаты
  (:predicates (clear ?l - location)
      ;; -- свободна ли локация
      (at ?t - thing ?l - location)
      ;; -- находится ли предмет в локации
      (at-goal ?s - stone)
      ;; -- находится ли камень в целевой локации
      (IS-GOAL ?l - location)
      ;; -- является ли локация целевой и т.д.
      (IS-NONGOAL ?l - location)
      (MOVE-DIR ?from ?to - location ?dir - direction))


  (:functions (total-cost) - number)
      ;; total-cost -- функция без аргументов,
      ;; возвращает одно число -- суммарную стоимость
      ;; примененных функций

  ;; далее следуют описания действий
  (:action move
      ;; действие -- двигаться
   :parameters (?p - player ?from ?to - location ?dir - direction)
```

```
   ;; четыре параметра, ограничение на типы

:precondition (and (at ?p ?from)
                   (clear ?to)
                   (MOVE-DIR ?from ?to ?dir)
                   )
   ;; предусловия -- игрок находится в локации ?from,
   ;; локация ?to свободна, а направление перемещения
   ;; из локации ?from  в ?to совпадает с ?dir
:effect       (and (not (at ?p ?from))
                   (not (clear ?to))
                   (at ?p ?to)
                   (clear ?from)
                   )
    ;; эффект от применения действия --
    ;; ?p не находится в локации ?form,
    ;; ?to больше не свободна,
    ;; ?p находится в локации ?to,
    ;; локация ?from теперь свободна
)

(:action push-to-nongoal
    ;; действие -- продвинуть камень в не целевую локацию
:parameters (?p - player ?s - stone
             ?ppos ?from ?to - location
             ?dir - direction)
:precondition (and (at ?p ?ppos)
                   (at ?s ?from)
                   (clear ?to)
                   (MOVE-DIR ?ppos ?from ?dir)
                   (MOVE-DIR ?from ?to ?dir)
                   (IS-NONGOAL ?to)
                   )
:effect       (and (not (at ?p ?ppos))
                   (not (at ?s ?from))
                   (not (clear ?to))
                   (at ?p ?from)
                   (at ?s ?to)
                   (clear ?ppos)
                   (not (at-goal ?s))
                   (increase (total-cost) 1)
                   )
)
<...>
)
```

Описание задачи на языке PDDL для данной предметной области представлено ниже с вырезанными фрагментами, так как оно требует описания **всех** объектов, их отношений между собой **со всеми** деталями и т.д., что может занять довольно много места:

```
;;   #######
;; # #     #
;; # # # # #
;;   # @ $ #
;; ### ### #
;; #   ### #
;; # $  ##.#
;; ## $  #.#
;;  ## $  .#
;; # ## $#.#
;; ## ## #.#
;; ### #   #
;; ### #####

(define (problem p109-microban-sequential)
  ;; определяем задачу

  (:domain sokoban-sequential)
      ;; указываем предметную область, для которой
      ;; сформулирована задача

  (:objects
    ;; перечисляем все объекты,
    ;; кторые фигурируют в задаче

    dir-down - direction
    dir-left - direction
    dir-right - direction
    dir-up - direction
    player-01 - player
    pos-01-01 - location
    pos-01-02 - location
    pos-01-03 - location
    pos-01-04 - location
    <..>
  (:init
    ;; формулируем начальное состояние системы
    ;; задаем значение предикатов на объектах системы
    (at player-01 pos-05-04)
    (at stone-01 pos-07-04)
    (at stone-02 pos-03-07)
    <..>
    (IS-GOAL pos-08-07)
    (IS-GOAL pos-08-08)
    (IS-GOAL pos-08-09)
    (IS-GOAL pos-08-10)
    (IS-GOAL pos-08-11)
    (IS-NONGOAL pos-01-01)
    (IS-NONGOAL pos-01-02)
    (IS-NONGOAL pos-01-03)
    (IS-NONGOAL pos-01-04)
    <..>
    (MOVE-DIR pos-01-01 pos-02-01 dir-right)
```

```
    (MOVE-DIR pos-01-04 pos-02-04 dir-right)
    (MOVE-DIR pos-02-01 pos-01-01 dir-left)
    <..>
    (clear pos-01-01)
    (clear pos-01-04)
    (clear pos-01-09)
    (clear pos-02-01)
    <..>
  )

  ;; теперь  формулируем ограничения на целевое состояние
  ;; все камни должны располагаться в целевых локациях
  (:goal (and
    (at-goal stone-01)
    (at-goal stone-02)
    (at-goal stone-03)
    (at-goal stone-04)
    (at-goal stone-05)
  ))

  ;; формулируем метрику, с помощью которой
  ;; оценивается качество планов
  (:metric minimize (total-cost))
)
```

PDDL имеет широкие возможности для описания знаний о моделях мира и задачах планирования. Он развивается и по сей день, в последнее время двигаясь в направлении представления знаний, близкому к объектному.

Объектное представление – представление, которое позволяет оперировать понятиями объекта, связанных с ним данных и методов, как с единым целым. Сейчас наряду с текстовыми языками представления знаний используются и графические языки, например UML [4] – Unified Modeling Language *(унифицированный язык моделирования)*. Данный язык хорошо подходит для записи знаний в объектном представлении.

Язык UML был создан для удовлетворения нужд программной индустрии, но его возможности оказались шире, поэтому он используется и в других областях. Например, его можно использовать для представления знаний систем планирования в виде UML моделей. Для создания UML моделей существует множество редакторов, что делает возможность задания знаний среды и задачи планирования в объектном представлении более привлекательной.

В данной дипломной работе рассматривается следующая задача: по

текстовым описаниям на языке PDDL требуется сгенерировать соответствующие им UML модели. Получение UML-представления может быть полезно, так как:

- UML модели более наглядны, составлять UML базы знаний могут эксперты, не знакомые с PDDL и его устройством;
- UML модель может быть отредактирована в графическом редакторе, затем оттранслирована в PDDL [5];
- можно использовать инструменты UML-валидации и исполнения для проверки корректности знаний [6, 7];
- по UML-описанию предметной области могут быть сформулированы задачи для этой предметной области [8].

Таким образом, предполагается создать программное средство, которое будет автоматизировать преобразование PDDL-описаний предметных областей и задач планирования в UML-модели. На вход средство будет получать PDDL-описания. Знания в этом представлении состоят из описания предметной области (типов, предикатов, функций, операций, а также их предусловий и эффектов), и описаний задач, каждая из которых представлена в виде описаний начальных и конечных состояний, требований предъявляемых к решению задачи. Создаваемое средство должно преобразовывать данное представление знаний, и на его выходе должно получиться объектное представление тех же знаний. В ходе работы средства к представлению будет применяться ряд преобразований. Пусть T – описываемое преобразование, $T : A \to B$, A – база знаний в исходном (текстовом PDDL-представлении), B – база знаний в целевом представлении (в виде UML-модели). Преобразование должно обладать следующими свойствами:

1. Любой PDDL-тип $C \in A \to C' \in B$, где C' – UML-класс (Рис. 1);

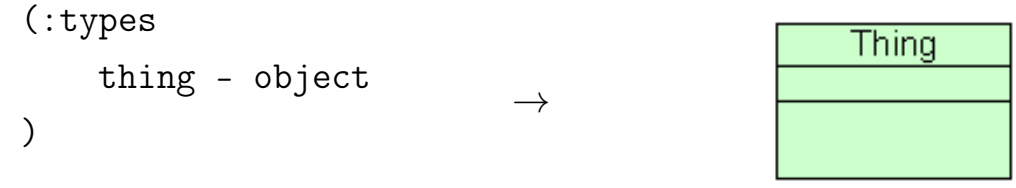

Рис. 1: Пример преобразования типов

2. Отношения между PDDL-типами $C_1, C_2 \in A$ должны переводиться в отношения между UML-классами $C'_1, C'_2 \in B$, с использованием атрибутов, ассоциаций и др. (Рис. 2);

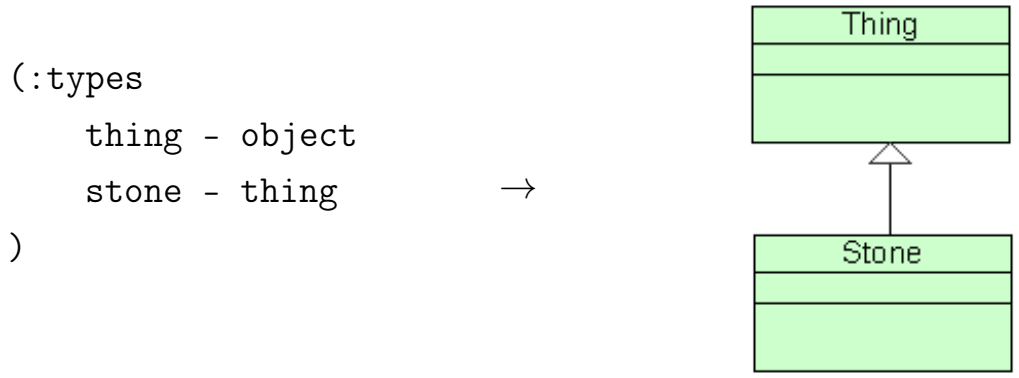

Рис. 2: Пример преобразования отношений

3. Любой PDDL-объект $o \in A \rightarrow o' \in B$, где o' – экземпляр UML-класса и если $C \in A$ – некоторый класс, то $C \rightarrow C' \in B, \Rightarrow o' \in C'$;

4. Любое PDDL-состояние $S \in A \rightarrow S' \in B$, где S' – состояние, описываемое UML. Причем:

   (a) Для любого PDDL-объекта в этом состоянии $o \in S$, то в S' должен существовать объект $o' \in B$, o' - образ объекта o;

   (b) Если PDDL-объекты $o_1, o_2 \in A$ связаны конструкциями, то их образы $o'_1, o'_2 \in B$ должны быть связаны соответствующими конструкциями, если это возможно;

5. Любое действие $a \in A$ имеет образ $a' \in B$, причем если в результате применения действия a в состоянии $s_1 \in A$ получается $s_2 = apply(a, s_1), s_2 \in A$, то образы $s'_1 \in B$ и $s'_2 \in B$ связаны аналогичным соотношением $s'_2 = apply(a', s'_1)$ (Рис. 3);

```
(:action move
:parameters (?p - player ?from ?to - location ?dir - direction)
:precondition (and (at ?p ?from)
                   (clear ?to)
                   (MOVE-DIR ?from ?to ?dir)
                   )
:effect       (and (not (at ?p ?from))
                   (not (clear ?to))
                   (at ?p ?to)
                   (clear ?from)
                   )
)
```

↓

+

OCL [9] ограничения на метод

«pre»

```
this.at == from and
to.isClear and
LocDirRelation->
exists(LocDirRelation{from, to, dir})
```

«effect»

```
! this.at == from and
! to.isClear and
this.at == to and
from.isClear
```

Рис. 3: Пример преобразования действий

6. PDDL-представление любой задачи (начальное состояние и цель) должно переводиться в соответствующее UML-представление с сохранением свойств объектов и отношений между ними, как описано в предыдущих пунктах.

Одной из сред поддержки процессов, связанных с планированием, является среда itSIMPLE [10]. В данной среде была решена обратная задача:

трансляция UML моделей в PDDL описания. Для этого было предложено использовать специальную семантику некоторых UML-элементов. Так или иначе, средство не решает задачу, поставленную в данной работе.

Подавляющее большинство инструментов, реализующих преобразование каких-либо текстовых данных в UML-модели, существует для работы с языками программирования, как правило, объектно-ориентированными, а не с языками представления знаний. Поэтому, использование этих инструментов для решения поставленной задачи невозможно.

Среди инструментов, работающих не только с языками программирования, можно выделить MoDosco. Данный инструмент позволяет создавать модели различных систем, используя специальные Discoverer'ы, которые на данный момент существуют в основном для Java-кода и JAR-архивов. Кроме того, данный проект находится в инкубаторе и о создании PDDL-Discoverer'ов говорить не приходится.

Литература

1. **Рассел С., Норвиг П.** *Искусственный интеллект: современный подход, 2-е изд.* М. : Вильямс, 2006. 1408 с.

2. **Fikes R., Nilsson N.** *STRIPS: a new approach to the application of theorem proving to problem solving // Artificial Intelligence.* 1971. 2. P. 189-208.

3. **Gerevini, A., Long, D.** *Plan constraints and preferences in PDDL3.* Technical Report, Univ. Brescia, Italy, 2005. 12p.

4. **Рамбо Дж., Блаха М.** *UML 2.0. Объектно-ориентированное моделирование и разработка, 2-е изд.* СПб. : Питер, 2007

5. **Малышко В. В., Манжосов А. В.** *Решение задач инженерии знаний средствами объектно-ориентированной инженерии программного обеспечения // Программные системы и инструменты. Тематический сборник No 13.* М. : Изд-во факультета ВМиК МГУ, 2012. стр. 44-54.

6. **Gogolla M., Büttner F., Richters M.** *USE: A UML-Based Specification Environment for Validating UML and OCL.* Science of Computer Programming, vol. 69, no. 1-3. 2007. pp. 27-34.

7. **Гладкова О. А.** *Визуализация и валидация планов при помощи объектных моделей.* Дипломная работа. М. : МГУ ВМК, кафедра системного программирования, семинар «Планирование целенаправленной деятельности», 2013.

8. **Долотказин Ю. В.** *Генерация описаний задач планирования по объектной модели предметной области.* Дипломная работа. М. : МГУ ВМК, кафедра системного программирования, семинар «Планирование целенаправленной деятельности», 2013.

9. **Warmer, Kleppe.** *The Object Constraint Language: Getting Your Models Ready for MDA, Second Edition.* Addison-Wesley, 2003.

10. **Vaquero T. S., Tonaco R. et al.** *itSIMPLE4.0: Enhancing the Modeling Experience of Planning Problems.* Proceedings of the ICAPS 2012 System Demonstration, São Paulo : s.n., 2012. pp. 11-14.