

# 2017 中国机器人大赛比赛规则

## 服务机器人项目 仿真项目

2017 中国机器人大赛服务机器人项目技术委员会

2017 年 5 月 31 日

## 目 录

一、项目简介 .....	3
二、技术委员会 .....	5
三、赛项说明 .....	5
3.1 概况 .....	5
3.2 场景描述 .....	6
3.3 机器人物理行动 .....	9
3.4 人机交互和观察行动 .....	12
3.5 任务描述 .....	13
3.5.1 指令交互项目 .....	13
3.5.2 自然语言交互项目 .....	16
3.6 比赛平台 .....	19
3.7 平台依赖 .....	22
3.8 平台接口 .....	22
3.8.1 场景描述格式 .....	22
3.8.2 任务描述格式 .....	23
3.8 平台运行以及调试 .....	24
四、比赛场地及器材 .....	25
五、评分标准 .....	26
六、赛程赛制 .....	28

附录 A.....	32
附录 B.....	32

## 一、项目简介

服务机器人仿真比赛立足于面向室内环境的服务机器人的高层功能的探索，主要包括人机交互、自动规划、推理、环境感知和重新规划。为此，将服务机器人抽象为仿真机器人，并以仿真的室内环境为测试环境，将人机交互抽象为自然语言或命令语言表达的任务描述和人机对话，将机器人感知数据抽象为文件格式的场景描述、观察反馈和执行反馈。

服务机器人仿真比赛针对自主机器人在家庭环境中的典型应用来设置一系列场景，以测试参赛程序的性能。具体说来，一开始仅被告知部分的 环境初始状态和需完成的任务，参赛程序可以通过人机交互（向平台提 问）或观察环境的行动以获取缺乏的信息，以便规划出行动序列来完成任 务。但初始状态信息和人机交互结果有可能是错误的，从而出现规划结果 中某行动无法执行的情况。此时，参赛程序会获得平台反馈的失败结果，并据此进行重新规划。其中，人机交互过程分别用指令语言和自然语言表 达，从而构成本赛事的两个不同的比赛项目——指令语言交互项目和自然 语言交互项目。

比赛基于一个 3D 仿真机器人，它有一组固定的原子行 (primitive actions)，包括九种物理行动、人机交互行动和观察行动，对所有问题都不变。本次仿真赛采用的仿真机器人如图 1.1 所示，考虑基于其功能的机器人问题求解。此机器人有两个轮子（可以

移动），手臂上有一个手爪（一次只能抓取一个东西），和一个盘子（上面只能放一个物体），体现基本的移动、抓取、放下、开门、关门等能力。同时，机器人可以向环境中用户提问，实现人机交互，也可以观察自身周围环境。在这些功能基础上，测试机器人的人机交互、自动规划、推理、观察、重新规划等能力。

服务机器人仿真比赛要求参赛程序对比赛平台提供的每一个场景，根据其初始状态描述和任务描述，在规定时间内，通过人机交互或观察行动获取必要的环境信息，自动规划出物理行动序列以实现目标，并根据行动反馈，调整规划结果，保证最终完成任务。比赛平台将根据参赛程序在此过程中的整体性能打分，并根据一个阶段中所有问题的总分决定参赛程序在该阶段的排名。

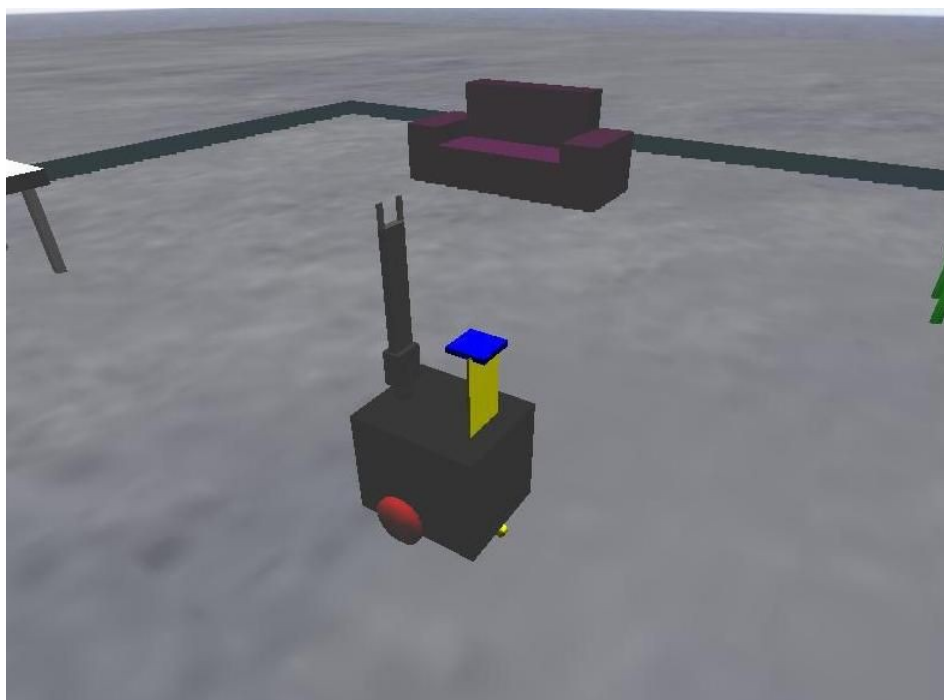


图 1.1: 机器人模型

## 二、技术委员会

负责人：刘江川，中国科学技术大学， [jkd@mail.ustc.edu.cn](mailto:jkd@mail.ustc.edu.cn)，  
18715000339

成 员：毛登辉，上海大学，

柴劲松，安徽大学

李冉冉，合肥师范学院

陶余钱，安徽建筑大学

## 三、赛项说明

### 3.1 概况

比赛过程中，机器人可选的物理行动、人机交互行动和观察行动是固定的。其中，物理行动包括移动、抓取、放下等机器人基本动作，每执行一个物理行动，平台会反馈其执行结果（成功或失败）。人机交互行动用来模拟机器人向用户提问，目前只有一个行动，问某具体物体所在的位置。平台根据一定概率，有三种可能的回答——正确位置，错误位置和不知道，并且对同一个问题，始终只有一种特定的回答。观察行动用来模拟机器人通过传感器对自身位置附近的观察，假设观察行动能够正确的告诉机器人当前所在位置的所有物体。

每个测试都基于一个完整的环境场景。一开始，参赛程序将获得环境的部分初始状态，包括环境中出现的所有物体，环境中所有大物体的位置信息，机器人状态信息，和部分小物体位置信息。其中，小

物体位置信息有一定概率会出错。同时，参赛程序也被告知需要完成的多个任务。其中，完成任务所需的一些位置信息，可能事先不知道或有错误，参赛程序可以通过人机交互行动或观察行动获取相关信息，也可以盲目搜索整个环境。参赛程序每执行一个行动，平台将反馈执行结果。目前，只有一种情况下行动会执行失败，当机器人要抓取物体，而不在当前位置时，此行动执行失败。此时，参赛程序可以通过人机交互行动或去其他位置观察来获取的位置，重新规划，以最终完成所有任务。

### 3.2 场景描述

场景描述规定当前机器人所面临的环境状态，即当前环境中出现的物体及其属性，包括物体的种类、位置和其他属性，以及机器人的状态等。

此次仿真比赛中，场景描述的内容包括：

场景大小：在整个比赛中保持不变（10×10 米）。

场景中物体描述：用编号表示不同的物体，对每个编号有相应的描述，包括物体的种类、位置和其他属性。这些描述都是可选的，有些物体可能没有完整的描述。其中有一个特殊的物体——人（sort: human），代表用户。比赛提供的所有场景中，有且仅有一个人（以下称为用户），并且在每一个任务的求解和执行过程中其状态不会改变。其他物体的具体描述内容包括：

sort: 物体的种类, 例如: book, cup, bottle, table, chair, microwave 等。物体种类清单在附录 A.1 中给出。

color: 物体的颜色: white, red, green, yellow, blue, black

size: 物体的类型: big, small。small 和 big 用来区分物体是否可以被机器人移动, 据此物体可以分为两类:

\* “大物体”: 机器人无法移动的物体, 由于没有外部影响, 它们的位置和状态在一个给定的问题中不会改变。“大物体”包括“小物体”可以放在上面的家具, 如 table, chair 等, 也包括“小物体”可以放在里面的家具或家电, 如 closet, refrigerator, microwave 等。

\* “小物体”: 机器人可以移动的物体, 位置可以改变, 通常在“大物体”上面或里面。

location: 为整数数值, 代表不同的位置编号, 并且假设, 任意不同的两个位置之间是可达的, 即机器人可以从任意位置移动到目标位置。一个位置内, 可以同时放置多个物体, 但至多只能有一个“大物体”。同时, 机器人通过观察行动可以获得当前位置上的所有物体, 但不包括在其他物体里面并且门没有开的“小物体”。

type: 目前需要额外说明的类型只有一种 container, 家庭环境中 closet, refrigerator, microwave等都是 container。

这里假设每个容器都有门可以开关, 同时小物体可以放在容器中, 也可以从容器中取出。具体对应的状态包括:



inside: 表示“小物体”A（物体编号）在“大物体”B 里面。

opened: 表示“大物体”的门处于开的状态。

closed: 表达“大物体”的门处于关的状态。

机器人状态：机器人有一个手爪，一个盘子，它们上面是否有东西也决定机器人的状态。具体的内容包括：

location: 机器人所在的位置（用 0 表示机器人），与物体位置的表达相同，机器人手爪和盘子上的东西也和机器人有同一位置。

plate: 机器人盘子的状态，赋值为空（用 0 表示）或者某个“小物体”的编号。

hold: 机器人手爪的状态，赋值为空（用 0 表示）或者某个“小物体”的编号。

比赛中所用的场景描述以文件形式存储，比赛平台拥有完整的场景描述，而参赛程序在一开始只能获得部分的场景的描述（缺乏一些小物体的位置信息），并且有一定概率出错（小物体位置有可能出错）。参赛程序可以通过人机交互行动、观察行动和行动反馈更新自己的场景描述。

问题求解往往不仅初始状态描述，还需要一些领域知识，例如，一个东西只能有一个位置，机器人盘子中的东西会随着机器人移动。这些知识需要参赛者自己添加。

这里只给出了场景描述的文字描述，相应的可以用 Planning Domain Definition Language (PDDL)描述，参见附录 B.2。比赛中

场景描述将以文件形式按一定格式提供给参赛程序，具体细节在比赛平台中介绍。

### 3.3 机器人物理行动

机器人物理行动，体现机器人的基本行动能力，是仿真平台对机器人底层功能的抽象。物理行动由其执行条件和执行效果来刻画，执行条件为一组对场景的约束，执行效果为对场景的改变。我们假定，机器人只能根据这些物理行动来改变场景状态。

参赛程序需要根据输入的问题，在规定时间内输出规划结果，即原子行动的序列，并根据平台反馈结果不断调整。其中，比赛平台会反馈每个物理行动是否执行成功，当某行物理动执行失败后，参赛程序需要根据反馈和当前状态重新规划，并将重新规划的结果（行动或行动序列）发给平台执行。若参赛程序发送了两个行动，第一个执行失败了，参赛程序重新发送第三个行动，则平台执行第三个行动，同时忽略第二个行动。最终，平台的评价程序将对最终的执行效果与需要完成的任务做对比，并根据任务完成情况给规划结果打分。具体评分原则在后面章节介绍。

本次仿真比赛中，机器人的物理行动为下列九种，在整个比赛中不会改变：

`move(X)`：机器人移动到位置  $X$ 。执行条件为机器人不在此位置（我们假设，地图中标出的其他位置都是可达的）。执行效果为机器人

移动到了位置 X（不再在原来的位置），不受此行动影响的其他状态属性不变（如果机器人手爪拿着东西，或者机器人盘子里面有东西，则这些东西也会随着机器人一起移动）。

`pickup(X)`：机器人拿起物体 A。执行条件为：A 在机器人的可取范围内（机器人与 A 在同一个位置内，并且不在机器人盘子上），A 可以拿的动（A 为小物体），不在其他物体（例如 refrigerator）内部，机器人手爪没有拿其他东西。执行效果为机器人手爪拿起了物体 A（手爪不再空），其他状态属性不变。

`putdown(X)`：机器人放下物体 A。执行条件为机器人手爪正拿着物体 A。执行效果为机器人手爪不再拿任何东西，其他状态属性不变（A 的位置仍然在机器人当前所在的位置）。

`toplate(A)`：机器人将 A 物体放入自己的盘子中。执行条件为机器人手爪拿着 A，盘子中没有其他物体。执行效果为机器人手爪不再拿任何东西，A 在盘子中，其他状态属性不变（A 的位置仍然在机器人当前位置，并且 A 以后会随机器人移动）。

`fromplate(A)`：机器人将物体 A 从盘子中拿起。执行条件为 A 在机器人盘子中，机器人手爪没有拿其他物体。执行效果为 A 不再在盘子中，机器人手爪拿着 A，其他状态属性不变。

`open(B)`（必须为 container，下同）：机器人打开容器 B 的门。执行条件为机器人和 B 在同一个位置，机器人手爪为空，并且 B 的门是关闭的。执行效果为 B 的门被打开(`opened(B)`)并且不再关闭

，其他属性不变。

`close(B)`：机器人关上容器 `B`。执行条件为机器人和 `B` 在同一个位置，机器人手抓为空，并且 `B` 的门是打开的。执行效果为 `B` 的门关闭 (`closed (B)`)不再打开，其他属性不变。

`putin(A ,B )`：机器人将小物体 `A` 放到容器 `B` 中。执行条件为机器人和 `A`在同一个位置，机器人手抓正拿着物体 `A`，并且 `B` 的门是打开的。执行效果为，小物体 `A` 在 `B` 的内部(`inside (A ,B )`)，手爪为空，其他状态属性不变。注意，这里不考虑容器的大小，即假设容器里面可以放任意多的小物体。

`takeout(A ,B )`：机器人将小物体 `A` 从 `B` 中取出来，执行条件为机器人和容器 `B` 在同一个位置,`A` 在 `B` 中，机器人手抓为空，并且 `B` 的门是打开的。执行效果为，机器人抓着小物体,`A` 不再处于 `B` 的内部，其他状态属性不变。

这里只给出了对这 9 种物理行动的文字描述，相应的形式化描述用 PDDL语言在附录 B.1 中给出。同时机器人还有人机交互行动和观察行动，在下一节介绍。

参赛程序每提交（执行）一个物理行动，平台将返回此行动是否执行成功。当某行动的执行条件没被满足时，平台返回执行失败。例如，执行行动`pickup(A)`，但由于错误信息或其他原因，`A` 并不在当前位置，则此行动执行失败。执行行动 `putin (A ,B )`，但容器的门是关的 `closed(B)`，则此行动也执行失败。

这些物理行动的具体接口在下面具体规定。

### 3.4 人机交互和观察行动

人机交互行动用来模拟机器人与用户的交互过程，这里只简单抽象为机器人对用户的提问，然后平台回答。

本次仿真比赛中，只有一种人机交互行动askloc(num)，用来询问num指代的物体所在的位置。注意，这里 num 需要是场景中一个“物体”的数字编号，否则平台返回空结果。在 num 合法的情况下，平台针对askloc(num)行动返回以下情况：

以 60% 的概率，返回小物体 obj 正确的位置关系，即 at(num, loc)或 inside(num ,num' ' )，其中 loc 为环境中的位置编号，num' ' 为相应的其他物体；

以 30%的概率，返回 obj 错误的位置关系，同样为 at(num ,loc)或inside(num ,num' ' )；

以 10%的概率，返回“不知道”（“not\_known”）。

机器人可以通过观察行动获得机器人当前位置上可观察到的所有物体。由于观察行动模拟来自机器人底层的传感信息，所以直接返回观察到 的物体编号。具体说来，机器人在任何时刻都可以执行观察行动，执行此行动，平台将返回：在机器人当前所在位置的所有物体，满足不在机器人手爪或盘子上，并且不在一个门是关着的容器里面。

### 3.5 任务描述

任务交互也是人机交互中的一个重要方面。在仿真比赛中，参赛程序 需要利用用户提供的任务描述，计算出完成任务的行动规划。

任务描述分为三种类型：目标、信息和约束。用户给定的目标代表用户要求机器人完成的工作，对应于希望达到的目标状态。例如，“Give me the green bottle.”，表示用户希望最终 “green bottle” 与用户在同一位置并且不在机器人身上。约束表示完成给定任务过程中必须遵守的约束条件，或者环境所满足的约束条件，例如 “Do not pick up any bottle on the table.”，表示在整个执行过程中，机器人都不能拿 “table” 上的 “bottle”。用户还可以提供关于初始场景的一些补充信息，例如 “The red bottle is on the table.” 即使在初始场景描述中没有规定 “red bottle” 的位置，也可以根据这条信息和 “table” 的位置推出 “red bottle” 的位置。

任务描述用两种方式表达，相应的挑战分为两个项目，其中一个用词汇受限的自然语言来表达任务描述，另一个用一些确定的指令表达。

#### 3.5.1 指令交互项目

此项目中，用户通过下面的指令集来表达相应的任务描述。指令中 sort 定义为场景描述中物体种类清单中的任意种类(附录

A.1)。指令中物体 `obj` 定义如下：

```
adj := big | small | white | black | red | green | yellow |  
blue  
obj := sort | adj sort
```

指令中任务目标 `task` 定义如下 (`obj1` 与 `obj2` 为节 2.4 中定义的物体)：

```
task := give(human, obj1) | puton( obj1, obj2) | goto(obj1)  
| putdown( obj1) | pickup(obj1) | open(obj1) | putin(  
obj1,obj2) | close(obj1) | takeout( obj1,obj2)
```

指令中补充信息 `info` 定义如下：

```
info := on(obj1,obj2) | near(obj1,obj2) | plate(obj1) |  
inside(obj1, obj2) | opened( obj1) | closed( obj1)
```

指令中约束 `cons` 定义如下：

```
cons := not task | not info | not not info
```

指令 `ins :=task | info | cons`。比赛中用到的指令集是 `ins` 的集合下面给出这些指令的含义。`task` 含义如下：

`give(human,obj )`：表示将物体 `obj` 给用户。`obj` 是“小物体”，要求`obj` 最终在用户的位置，并且它不在机器人的手爪中或盘子上。

puton(obj1,obj2): 将物体 obj1 放到 obj2 上面。obj1 是“小物体”，要求obj1 在 obj2 所在的位置(obj2 位置不变)，并且它不在机器人的手爪中或盘子上。

goto(obj): 机器人移动到 obj 所在的位置。

putdown(obj): 机器人放下物体 obj。obj 是“小物体”，要求obj 不再在机器人的手爪中或盘子上。

pickup(obj): 机器人拿起物体 obj。obj 是“小物体”，要求机器人的手爪握住 obj 或者 obj 在盘子上。

open(obj): 机器人打开容器 obj 的门。其中 obj 为 container，obj 的门被打开，即 opened(obj)为真。

putin( obj1, obj2): 机器人将小物体 obj1 放到容器 obj2 中，要求obj1在 obj2 内部，即 inside(obj1,obj2)为真。

close(obj): 机器人关闭容器 obj 的门。其中 obj 为 container，obj的门被关闭，即 closed(obj)为真。

takeout(obj1, obj2): 机器人将小物体 obj1 从容器 obj2 中取出，即inside( obj1, obj2)不再为真。

info 含义如下：

on(obj1, obj2): 表示物体 obj1 在 obj2 上面。解释为在场景中，



`obj1` 与 `obj2` 在相同的位置，并且 `obj1` 不在机器人上。

`near(obj1, obj2)`：表示物体 `obj1` 在 `obj2` 旁边。解释为在场景中，`obj1`与 `obj2` 在相同的位置。

`plate(obj)`：解释为物体 `obj` 在机器人盘子上。

`inside(obj1, obj2)`：表示小物 `obj1` 在容器 `obj2` 里面。

`opened(obj)`：表示容器 `obj` 的门处于打开状态。

`closed(obj)`：表示容器 `obj` 的门处于关闭状态。

`cons` 含义如下：

`not task`：表示在环境的任何状态(执行某个行动序列后的状态)，都禁止执行 `task` 所规定的行动。

`not info`：表示在环境的任何状态，`info` 所描述的情况都禁止出现。

`not not info`：表示在环境的任何状态，`info` 所描述的情况都必须保持。

### 3.5.2 自然语言交互项目

自然语言交互项目中，用户用词汇受限的英语来表达任务描述。词汇表虽然受限，但运用这些词汇仍然可以构造出复杂的句子和多种等价的表达。例如：“Give me the red bottle which is near

the green bottle.”, “The white cup is on the table.”, “The cup which is white is on the table.”, “There must be a bottle on the desk.”.

比赛中自然语言可以出现的词汇如表 2.1 (Table 2.1) 所示。

表 2.1: 自然语言词汇表

词汇类型	词汇
代词	me
冠词	a, an, the
助动词	must
动词	do, be, give, put, go, pick, take, open, close
介词	in, out, on, near, next, to, up, down, from, of, which
形容词	white, black, red, green, yellow, blue, big, small, each, every, all
副词	there, not
名词	plate, door, 所有物体的种类名称

可以看到 Table 2.1 中词汇非常有限, 针对应用背景避免了一些复杂情况。代词只有 me, 特指场景中用户(场景中, 人是唯一的), 没有其他代词, 从而避免了自然语言中复杂指代关系的分析。介词中增加 which, 从而可以表达常用的定语从句。形容词来自于场景中对物体属性的描述。名词 plate 特指机器人身上的盘子, 用户可以给出任务 “Put the red bottle on the plate.”。一般 “must”, “do not” 用来表达约束。

上面词汇的语法按照英语的一般语法标准, 这里不会特别限制。比赛中, 用这些词汇构成的自然语言语句来表达任务描述。

对于任务描述中的目标, 一般用祈使句表达, 例如: “Go to the table.”, “Pick up the bottle which is on the chair.”。其

中主动动词含义解释如下（解释为对物体位置和机器人状态的要求，其 A,B 指某个物体）：

give: 一般用法为 “give somebody A ” 或者 “give A to somebody”。作为任务，要求最终物体 A 的位置在 “somebody” 所在的位置，并且它不在机器人的手爪中或盘子上。

put: 一般用法有三种，第一种 “put A on/near/next to/down to B ”，要求最终物体 A 在物体 B 所在的位置（B 位置不变），并且它不在机器人的手爪中或盘子上。第二种 “put A down” 或者 “put down A”，要求不再在机器人的手爪中。第三种 “put in ”，要求最终物体 在容器的内部。

go: 一般用法为 “go to A ”，要求机器人移动到 A 所在的位置。

pick: 一般用法为 “pick up A ”，要求机器人的手爪握住 A。

take: 一般用法为 “take out A from B ”，要求从容器 B 中取出 A。

open: 一般用法为 “open B” 或者 “open the door of B”，要求打开容器B 的门。

close: 一般用法为 “close B” 或者 “close the door of B”，要求关闭容器的门。

对于补充信息，用陈述句表达，例如：“The book is on the table.”。一般为 “A is on/near/next to B” 或者 “There is

A on/near/next to B” 的形式，它们都解释为 A 与 B 在同一个位置，除了 B 为 plate 的情况，此时解释为 A 在机器人的盘子里。又例如，“A can is in the microwave.”，表示一个 ‘can’ 在 ‘microwave’ 中。“The door of the microwave is opened”，表示 ‘microwave’ 的门是打开的。

任务描述中约束用 “Do not . . . ”， “There must (not) be . . . ”，或者 “. . . must (not) be . . . ” 的方式表达。“Do not” 后面一般接任务目标，例如 “Do not go to A ”， “Do not put A on B ”。“There must (not) be” 后面一般接信息描述，例如 “There must (not) be A on B ”。“ . . . must (not) be. . . ” 一般描述两物体的关系，例如 “The red bottle must (not) be on the plate”。

### 3.6 比赛平台

本次比赛使用 ChallengeServer 作为比赛平台，平台以及运行环境将公布在：

<http://wrighteagle.org/homesimulation>

ChallengeServer 负责管理比赛问题集（场景描述和任务描述），启动之后监听特定端口（默认是 7932），等待客户端连接进来，连接之后 server 负责向 client 发送问题的场景描述和任务描述文件。client 利用这些信息，在指定的时间（默认是 5s）之内进

行规划，规划之后将结果发送回 server，server 对其进行评价以及打分。为简单公平起见，运行时保证只有一个 client 连接。与以往不同的地方是平台将会加入在线反馈功能，执行动作后平台会返回执行效果。参赛队伍可以选择实现在线规划，规划一次行动，调用行动相应的接口函数得到反馈，按照反馈结果执行下一次行动的规划。

server 可以有各种启动参数，这些命令可以用 “./cserver - help” 查看，同时 server 也支持从网络连接。

参赛队需要自己实现 client 端的程序，编译成一个可执行文件并连接上 server 进行比赛。编写 client 函数具体可以见平台中的 example 文件夹。实现自己的规划器，首先需要包含头文件 `#include "cserver/plug.hpp"`，然后重载其中的一些函数，其接口说明见表 4.1。同时需要一个主函数，在其中调用 Run 方法，详细可以见平台中的 example 文件夹。

ChallengeServer 提供 SDK，参赛队需要按照 SDK 提供的接口函数，自己实现 client 端的程序，编译成一个可执行文件并连接上 server 进行比赛。表 3.1 给出了 ChallengeServer SDK 的接口函数说明。

挑战赛分为两个阶段，每个阶段都需要参赛程序求解一组问题集（场景描述和任务描述）。比赛所用的问题集并不事先给出，比赛结束后会统一发布。

表 3.1: 规划器客户端的接口函数列表

void Plan()	规划的主体代码，需要执行各规划动作
void Fini()	结束函数，每一次规划结束之后调用
string& GetName()	返回队伍的名字
string& GetEnvDes()	返回规划的环境描述字符串。
string& GetTaskDes()	返回规划的任务描述字符串。
bool Move(unsigned x)	执行动作 “move” 到地点 “x”， 返回行动失败还是成功
bool PickUp(unsigned a)	执行动作 “pickup” 小物体 “a”， 返回行动失败还是成功
bool PutDown(unsigned a)	执行动作 “putdown” 小物体 “a”， 返回行动失败还是成功
bool ToPlate(unsigned a)	执行动作 “toplate” 小物体 “a”， 返回行动失败还是成功
bool FromPlate(unsigned a)	执行动作 “fromplate” 小物体 “a”， 返回行动失败还是成功
bool Open(unsigned a)	执行动作 “open” 物体 “a”， 返回行动失败还是成功
bool Close(unsigned a)	执行动作 “close” 物体 “a”， 返回行动失败还是成功
bool PutIn(unsigned a,b)	执行动作 “putin” 小物体 “a”， 大物体 “b”，返回行动失败还是成功
bool TakeOut(unsigned a,b)	执行动作 “takeout” 小物体 “a”， 大物体 “b”，返回行动失败还是成功
string AskLoc(unsigned a)	执行人机交互动作 “askloc” 物体 “a”，当 不知道物体位置时，返回 “unknown”；当返 回位置关系时，为 “(on a b)”、“(near a b)” 或 “(inside a b)”， “b” 为相应其他的物体
void Sense(set& A)	执行观察行动，“A” 返回观察到的 所有物体的编号集合

为了方便调试，ChallengeServer 源代码 tests 文件夹包含一个场景描述和任务描述样例。

仿真平台的测试环境如下：

系统：Win7

CPU：待定...

Memory：待定...

### 3.7 平台依赖

目前在 Win7(Visual Studio 2010) 下面经过了测试。平台编译运行需要依赖的额外库和工具有

- Boost 1.51.0<sup>1</sup>
- CMake 2.8.9<sup>2</sup>

理论上在 Windows 其他版本的系统和编译器下面也能够正常编译运行运行。

详细的编译以及测试指南见平台压缩包 doc 文件夹下面的 plan-ner build.pdf 文件。

### 3.8 平台接口

在此平台上，为了便于程序与平台交流，采用 s-expressions<sup>3</sup> 来表达场景描述和（指令）任务描述。

#### 3.8.1 场景描述格式

比赛中，文件按下面的格式记录场景描述。首先，场景中不同的物体 和位置用正整数表示，分别记为<num>和<loc>。sort（附录 A.1）中某 一个物体名称记为<sname>，color 中给出的某个颜色名称记为 <cname>。用 BNF 范式定义场景描述的格式如下：

```
<sort> ::= "(sort" <num> <sname> ")"
<color> ::= "(color" <num> <cname> ")"
<size> ::= "(size" <num> "big)" | "(size" <num> "small)"
<type> ::= "(type" <num> "container)"
<location> ::= "(at" <num> <loc> ")" | "(at 0" <loc> ")"
<inside> ::= "(inside" <num> <num> ")"
```

---

```

<position> ::= "(on" <num> <num> ")" | "(near" <num> <num> ")"
<door> ::= "(opened" <num> ")" | "(closed" <num> ")"
<plate> ::= "(plate" <num> ")" | "(plate 0)"
<hold> ::= "(hold" <num> ")" | "(hold 0)"
<state> ::= <sort> | <color> | <size> | <type> | <location>
| <inside> | <position> | <door> | <plate> | <hold>
<domain> ::= "(:domain" <state>* ")"

```

其中 (at 0 <loc>) 表示机器人所在位置, (plate 0)和(hold 0)分别表示, 盘子为空和手爪为空。平台传递给参赛程序的场景描述为 s-expression<domain>, 例如:

```

(:domain (at 0 0) (plate 0) (hold 0) (sort 1 human) (at 1 1)
(size 1 big) (sort 2 table) (at 2 2) (size 2 big)
(sort 3 bottle) (color 3 green) (size 3 small) (at 3 2))

```

### 3.8.2 任务描述格式

对于指令交互项目, 按下面的格式记录任务描述。令  $i$  variable  $i$  为首字母大写的字符串, <predicate> 为将 <state> 中 <num> 和 <loc> 替换 为 <variable> 后的表达式。用 BNF 范式定义任务描述的格式如下:

```

<condition> ::= "(:cond" <predicate>* ")"
<tname> ::= "(give human" <variable> ")"
— "(puton" <variable> <variable> ")"
— "(goto" <variable> ")" | "(putdown" <variable> ")"
— "(pickup" <variable> ")" | "(open" <variable> ")"
— "(close" <variable> ")"
— "(putin" <variable> <variable> ")"
— "(takeout" <variable> <variable> ")"
<task> ::= "(:task" <tname> <condition> ")"
<info> ::= "(:info" <predicate> <condition> ")"
<cons> ::= "(:cons\_not" <info> ")" — "(:cons\_not" <task> ")"
| "("cons\_notnot" <info> ")"
<ins> ::= "(:ins" (<task> | <info> | <cons>)* ")"

```

其中<condition> 的作用是用来限制 <tname> 或 <predicate>



中<variable> 的取值范围。

平台传递给参赛程序的任务描述为 s-expression <ins>, 例如:

```
(:ins (:info (on X Y) (:cond (sort X bottle) (color X white) (sort Y table)))
(:task (give human X) (:cond (sort X bottle) (color X red))) (:cons\_`notnot
(:info (on X Y) (:cond (sort X bottle)
(sort Y table))))))
```

对于自然语言交互项目, 记录任务描述的文件为满足词汇表约束的英语语句的集合, 一般一句一行。例如:

```
The white bottle is on the
worktable. Give me the red bottle.
Put the blue bottle on desk.
There must be a bottle on the
table. Do not go to the teapoy.
```

### 3.8 平台运行以及调试

ChallengeServer 的 server 端提供二进制文件, 客户端文件可以自己调试同时由于程序采用多线程, 断点调试会影响程序本身的行为, 因此推荐大家采用在自己的程序中合适的地方输出相应的变量的值的方法进行调试。

具体的如何启动运行 server, 请看压缩包 doc 中的 planner\_build.pdf 文件。

平台 server 的具体参数如下:

```
cserver [-port digit] [-td dir] [-mode 'nt'—'it'] [-test id—'all'] [-to time]
[-cheat] [-log [dir]] [-eval [path [agr]]] [-help — -?]
```

-port num sets the port for the network conversation, default 7932

-td dir sets <dir> as the directory of the resource of test

-mode 'nt'—'it' sets challenge mode, nt or it  
nt means task is given via natural

language it means task is given via  
instruction

-test id runs the id th test runs all the test as default, or in the  
following condition id =all or id is out of range of the size of  
the test set

-to ms sets the period of timeout as ms

-cheat just cheat!

-log dir logs all info and the test result in <dir>  
<dir> is optional, default as the current working directory

-eval path agrs sets the evaluator in <path>  
<path> is default as asp and optional,  
empty <path> means no evaluator  
<args> is arguments for the evaluator  
Please use double quotes to contain the arguments

-help—? look for more commands

比赛中，参赛程序需要提交可以运行的二进制的客户端以及需  
要用的资源。

## 四、比赛场地及器材

仿真的场地需求为：

比赛区域长×宽为 5m×4m.

比赛器材：

台式计算机（2台）

计算机配置要求如下：

CPU：酷睿i7      内存容量：4GB      系统位数：64位

显卡：1024×768以上分辨率

操作系统： Ubuntu 12.04.2 LTS（或以上）

编译器 gcc 4.6.3

依赖： Boost 1.48（或以上）

Cmake 2.6.0（或以上） iclingo 3.0.5

投影仪（1台）

幕布（1套）

桌子（10张）

椅子（20条）

插线板（3个）

网线（2根）

## 五、评分标准

服务机器人仿真比赛要求参赛程序对比赛平台提供的每一个问题，根据初始的场景描述和任务描述，计算出合理的行动策略，规定由人机交互行动、观察行动和物理行动组成的行动序列，以完成任务，并且在执行失败后重新规划。参赛程序必须在规定时间内结束运行，平台将根据程序运行时间，已完成的任务数，不违反的约束数，和执行的行动数，综合计算出参赛程序的分数，并根据所有问题的总分对参赛程序排名次。

参赛程序完成任务描述的情况，由其完成的目标数目和维护的约束数目决定。一个任务描述可能含有多个目标或约束（不考虑补

充信息），自然语言中的一句话，指令表达中的一条指令，就表达一个目标或约束。行动序列完成一个目标或维护一个约束，定义如下：

参赛程序的终止状态：正常情况下，从场景描述的初始状态出发，参赛程序一个个的提交行动（包括人机交互行动，观察行动和物理行动）。这些行动有些执行成功，有些执行失败。对于物理行动，若执行成功，则场景状态被相应的改变；若执行失败，则场景状态保持不变。参赛程序应根据行动的反馈，合理调整行动策略。参赛程序结束运行时或超过规定时间进程中止时的场景状态，作为其终止状态。

参赛程序完成一个目标：其终止状态满足此目标的要求。

参赛程序维护一个约束：从初始状态到终止状态中间每一步的状态都满足此约束的要求。

评分标准如下：

完成一个目标，获得 40 分。

维护一个约束，获得 20 分。前提是完成至少一个目标，如果一个原子行动序列不能完成任何目标，则不计算其维护的约束的分数。

执行行动会扣除相应的分数，不论执行是否成功：

执行一次 move 行动扣除 4 分。

执行一次人机交互行动或其他物理行动扣除 2 分。

执行一个观察行动扣除 1 分。

从第一个行动开始记分，直到参赛程序结束运行或超时中止。所以，一个参赛程序对于一个问题的基础得分（此处  $\text{sgn}$  为符号函数）：

问题基础得分 =  $20 \times \text{维护的约束数目} \times \text{sgn}(\text{完成目标})$   
 $+ 40 \times \text{完成的目标数目} - 4 \times \text{move 行动个数} - 2 \times \text{人机交互行动} - 2 \times \text{其他物理行动个数} - \text{观察行动个数}$ 。

此次比赛在评分中，还考虑参赛程序的效率因素。在规定时间内，每提前 0.1s 增加 1 分（要保证至少完成一个目标）。具体说来，问题的效率得分（ $\lceil \cdot \rceil$  为取上整符号）：

问题效率得分 =  $2 \times \lceil (\text{规定时间} - \text{□□□□□□□□}) \times 10 \rceil \times \text{sgn}(\text{完成目标})$ 。

参赛程序对于一个问题的最终得分为：问题基础得分 + 问题效率得分。比赛的每个阶段都会给出一组问题，按当前阶段所有问题的总得分进行排名。如果有两支或多支队伍某阶段得分相同，则根据他们前一阶段的得分高低决定他们在此阶段排名中的相对名次。

## 六、赛程赛制

1、赛前会议：比赛前一天，技术委员会召集所有参赛队伍开负

责人开会。

2、调试：赛前会议召开完毕，即可进入调试阶段，调试分两阶段，各队自愿参与。

3、比赛：比赛成绩以第二阶段为准，具体如下：

本次比赛分为两个项目，指令交互项目和自然语言交互项目，每个项目包括两个阶段。每个阶段都包含一系列由场景描述和任务描述组成的问题（以及可能的错误信息）。参赛程序需要在固定的时间内，计算出可以完成指定任务的行动策略，包括何时采用人机交互和观察行动，如何用一组物理行动完成任务，并且在执行失败后重新规划。比赛平台将根据程序运行时间，已完成的任务数，不违反的约束数和执行的行动数，综合考虑，计算出参赛程序的分数，并根据每个项目第二阶段中所有问题的总分对参赛程序排名次。

另外请注意：第一阶段的问题中，机器人获得的初始状态一定是正确的，并且用户的回答一定是正确的。但参赛程序获得的初始状态可能不完全，并且任务描述中包含额外信息和约束。

下面介绍第一阶段的一个典型问题。已知完整场景初始状态为：

```
(:domain (at 0 1) (plate 0)    (hold 0)
      (sort 1 human) (at 1 1) (size 1 big)
      (sort 2 couch) (at 2 2) (size 2 big)
      (sort 3 table) (at 3 3) (size 3 big)
      (sort 4 cupboard) (at 4 4) (size 4 big)
                        (type 4 container) (closed 4)
      (sort 5 refrigerator) (at 5 5) (size 5 big)
                        (type 5 container) (closed 5)
```

```
(sort 6 book) (at 6 3) (color 6 red) (size 6 small)
(sort 7 can) (at 7 4) (color 7 red) (size 7 small)
(sort 8 can) (color 8 green) (size 8 small) (inside 8 5)
(sort 9 bottle) (at 9 2) (color 9 red) (size 9 small)
(sort 10 bottle) (at 10 2) (color 10 green) (size 10 small))
```

参赛程序获得的场景初始状态为

```
(:domain (at 0 1) (plate 0) (hold 0)
  (sort 1 human) (at 1 1) (size 1 big)
  (sort 2 couch) (at 2 2) (size 2 big)
  (sort 3 table) (at 3 3) (size 3 big)
  (sort 4 cupboard) (at 4 4) (size 4 big)
    (type 4 container) (closed 4)
  (sort 5 refrigerator) (at 5 5) (size 5 big)
    (type 5 container) (closed 5)
  (sort 6 book) (at 6 3) (color 6 red) (size 6 small)
  (sort 7 can) (color 7 red) (size 7 small)
  (sort 8 can) (color 8 green) (size 8 small)
  (sort 9 bottle) (color 9 red) (size 9 small)
  (sort 10 bottle) (at 10 2) (color 10 green) (size 10 small))
```

即，程序不知道 green can, red can 和 red bottle 的位置。参

赛程序同时获得的指令语言任务描述为：

```
(:ins (:task (puton X Y) (:cond (sort X can) (color X green)
  (sort Y table)))
  (:task (closed X) (:cond (type X
  container))) (:cons\_notnot
  (:info (inside X Y)
  (:cond (sort X can) (sort Y refrigerator))))))
```

相应的自然语言任务描述为：

Put the can which is green on the table. Close the door of each container.

There must be a can in the refrigerator.

一种可能的行动策略为：参赛程序先调用人机交互行

askloc(8)，由于第一阶段回答一定是正确的，获得平台反馈(inside 8 5)，机器人再提问 askloc(8)，得到回答 (at 7 4)，之后执行以下

物理行动:

```
move(4),pickup(7),move(5),toplate(7),open(5),fromplate(7),putin(7,5),take
out(8,5),toplate(8),close(5),move(3),fromplate(8),putdown(8).
```

上述行动策略可以完成所有目标，并且维护约束，相应的得分为:

$$40 \times 2 + 20 - 4 \times 3 - 2 \times 2 - 2 \times 10 = 64.$$

第二阶段的问题中，参赛程序获得的初始状态可能会出错，并且用户也有一定概率回答错误或不知道。下面，在上一节例子的基础上，介绍第二阶段的一个典型问题。

此问题与第一阶段问题的完整场景描述，参赛程序获得的初始场景描述和任务描述完全相同。只是在回答机器人提问时，答案不同。

一个具体的执行过程如下：参赛程序先调用人机交互行动 askloc(8)，此时平台回答 (inside 8 5) 是正确信息，机器人再问 askloc(7)，得到回答 (at 7 3)，是个错误答案。机器执行行动 move(3)，然后 pickup(7)，此时平台反馈执行失败。机器人此时只能盲目搜索，执行以下行动

```
move(1),sense,move(2),sense,move(4),sense,
```

最后一个观察行动返回观察到 7，机器人再执行以下动:

```
pickup(7),move(5),toplate(7),open(5),fromplate(7),putin(7,5),takeout(8,
5), toplate(8),close(5),move(3),fromplate(8),putdown(8).
```

上述行动策略可以完成所有目标，并且维护约束，相应的得分为:



$$40 \times 2 + 20 - 4 \times 6 - 2 \times 2 - 2 \times 11 - 3 = 50.$$

4、裁判员负责记录分数，比赛结束向大家宣布，若无异议，则签字。

5、技术委员会召开技术会议，总结此次比赛，并促进各队交流。

## 附录 A

补充材料

### A.1 物体种类清单

场景描述中物体的种类清单如下：

human(场景中唯一出现的用户), plant couch, chair, sofa, bed, table, workspace, worktable, teapoy, desk, television, airconditioner, washmachine, closet, cupboard, refrigerator, microwave, book, can, remotecontrol, bottle, cup.

## 附录 B

比赛问题的形式化描述

### B.1 物理行动的 PDDL 描述

```
(define (domain homerobot) (:requirements :strips :typing :equality
:existential-preconditions
:conditional-effects :domain-axioms)
(:types locs objs -number colors sorts) (:constants (0
-objs))
(:predicates      (at ?x - objs ?y - locs)
(inside ?x -objs ?y - objs) (hold ?x - objs) (handempty)
(plate ?x - objs) (plateempty) (samelocs ?x - objs) (opened ?x - objs)
(closed ?x - objs) (big ?x - objs) (small ?x - objs))
```

```
(color ?x - objs ?c - colors) (sort ?x - objs ?s - sorts)
)
```

```
(:action move
:parameters (?y - locs)
:precondition (not (at 0 ?y))
:effect
```

```
(and (at 0 ?y)
(forall (?z - locs)
(when (and (at 0 ?z) (not (= ?z ?y))) (not (at 0 ?z))))
(forall (?z - locs ?m - objs)
(when (and (at 0 ?z) (not (= ?z ?y)) (hold ?m))
(and (not (at ?m ?z)) (at ?m ?y)))) (forall (?n - objs)
```

```
(when (and (at 0 ?z) (not (= ?z ?y)) (plate ?n))
(and (not (at ?n ?z)) (at ?n ?y))))
)
```

```
(:action pickup
:parameters (?x - objs)
:precondition (and (small ?x) (handempty) (samelocs ?x))
:effect (and (not
(handempty)
) (hold ?x))
)
```

```
(:action putdown
:parameters (?x - objs)
:precondition (hold ?x)
:effect
(and (not
(hold ?x)) (handempty))
)
```

```
(:action toplate
:parameters (?x - objs)
:precondition (and (hold ?x) (plateempty))
:effect (and (not
(hold ?x)) (not
(plateempty)) (handempty) (plate ?x))
)
```

```
(:action fromplate
:parameters (?x - objs)
:precondition (and (plate ?x) (handempty))
:effect
(and (not (plate ?x))
(not (handempty)) (plateempty) (hold ?x))

)
(:action open
:parameters (?x - objs)
:precondition (and (samelocs ?x)
(handempty) (closed ?x))
:effect
(and (not closed ?x) (opened ?x) (handempty))
)
(:action close
:parameters (?x - objs)
:precondition (and (samelocs ?x)
(handempty) (opened ?x))
:effect
(and (not opened ?x) (closed ?x) (handempty))
)
(:action putin
:parameters (?x - objs ?y - objs)
:precondition (and (samelocs ?x)
(samelocs ?y) (opened ?y) (hold ?x))
:effects
(and (handempty) (opened ?y) (inside ?x ?y))
)
(:action takeout
:parameters (?x - objs y? - objs)
:precondition (and (samelocs ?x)
(samelocs ?y) (opened ?y) (handempty))

:effects
(and (hold ?x) (not
inside ?x ?y)
(opened ?y))
)

(:axiom
```

```
:vars (?x
-objs)

:context
(exists (?l -locs)
(and (at ?x ?l)

(at0 ?l)))
:implies (samelocs ?x)
)
)
```

## B.2 场景描述 PDDL 描述举例

```
(define (problem HOMESTATE1)
(:domain HOMEROBOT)
(:objects white red green yellow blue black - colors human couch cupboard
chair table teapoy book
```

```
can
```

```
remotecontrol refrigerator television bottle plant cup sofa airconditioner
workspace worktable bed desk - sorts)
```

```
(:init (at ROBOT 0) (plateempty) (handempty)
(sort 1 human) (at 1 1) (big 1)
(sort 2 couch) (at 2 2) (big 2)
(sort 3 table) (at 3 3) (big 3)
(sort 4 cupboard) (at 4 4) (big 4)
(sort 5 teapoy) (at 5 5) (big 5)
(sort 6 television) (at 6 6) (big 6)
(sort 7 worktable) (at 7 7) (big 7)
(sort 8 refrigerator) (at 8 8) (big 8)
(sort 9 desk) (at 9 9) (big 9)
(sort 10 book) (color 10 red) (at 10 7) (small 10)
(sort 11 can) (color 11 red) (at 11 4) (small 11)
(sort 12 can) (color 12 green) (at 12 5) (small 12)
(sort 13 bottle) (color 13 red) (at 13 3) (small 13)
(sort 14 bottle) (color 14 blue) (at 14 3) (small 14)
(sort 15 bottle) (color 15 green) (at 15 5) (small 15)
(sort 16 remotecontrol) (at 16 2) (small 16)
```

(sort 17 cup) (color 17 blue) (at 17 9) (small 17))  
(:goal (and (at 13 5) (at 11 5) (plateempty) (handempty)))