Chapter 10

Планиране

10.1 Лекции

10.1.1 Дефиниция на задачата за планиране

- Частен случай на търсене в пространството на състоянията при което се използува логическо представяне на целите и операторите.
- Състоянията имат логическа структура (не са черни кутии, както при търсенето), което дава възможност за ефективно планиране.
- Пример: планиране на действия на робот в "света на кубчетата" (фигура 10.1).

10.1.2 Situation Calculus

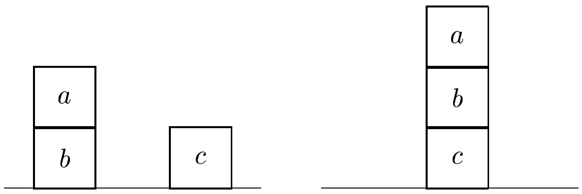
- Използуване на предикатното смятане за формализация на *ситуациите* и *действията* (McCarthy& Hayes, 1969).
- Действия и факти се представят чрез термове, например: $puton(A, B)$, $on(A, B)$.
- Верните факти в дадена ситуация се описват с предиката $holds$. Например $holds(on(A, B), S)$ означава, че A е върху B в ситуацията S.
- *Ситуациите* са термове, описващи състоянията на света. При зададена начална ситуация, например $s0$, всяка следваща се получава чрез функцията $result$. Например $result(puton(a, b), s0)$ представя ситуацията, получена при прилагане на действие $puton(a, b)$ в ситуация $s0$. По този начин можем да записване и по-сложни ситуации: $result(puton(c, a), result(puton(a, b), s0))$.
- Удобно представяне в Пролог: начална ситуация [], действието $puton(A, B)$ в ситуация S води до ситуация $[puton(A, B)|S]$. По този начин всяка ситуация директно описва *плана* (действията са в обратен ред) за достигането си.
- *Аксиомите* се представят с импликации, свързващи предусловия и ефекти, изразени чрез предиката $holds$. Например:

$$holds(clear(A), S) \wedge holds(clear(B), S) \rightarrow holds(on(A, B), result(puton(A, B), S))$$

- Тъй като по този начин ситуацията се променя, необходимо е да се постулира и кой факти остават в сила и в новата ситуация, т.е. *какво не се променя* при извършване на действието. Това става чрез т.нар. *фреймови аксиоми*. Например:

$$holds(clear(X), S) \wedge \neg eq(X, B) \rightarrow holds(clear(X), result(puton(A, B), S))$$

- За всеки един факт, който се използува при представянento трябва да се зададе съответна фреймова аксиома (frame problem).
- Неефективност при търсенето.



Начално състояние

$on(a,b)$
$on(b,table)$
$on(c,table)$
$clear(a)$
$clear(c)$

Целево състояние

$on(a,b)$
$on(b,c)$
$on(c,table)$
$clear(a)$

План: $puton(a,table), puton(b,c), puton(a,b)$

Figure 10.1: Планиране в "света на кубчетата"

10.1.3 Продход за планиране STRIPS

- Опит за решаване на фреймовия проблем (Fikes&Nilsson, 1971).
- Всяко действие се представя с три компонента:
  - *Предусловие*: формула, която трябва да е вярна за да се изпълни действието.
  - *Списък за добавяне*: формули, които стават верни след изпълнение на действието.
  - *Списък за изтриване*: формули, които стават невалидни след изпълнение на действието.
- Пример: $puton(X,Y)$
  - Precondition: $clear(X) \wedge clear(Y) \wedge on(X,Z)$
  - Add list: $\{on(X,Y), clear(Z)\}$
  - Delete list: $\{clear(Y), on(X,Z)\}$
- *Предположение*: всяка формула, която е била истина преди действието и не принадлежи на списъка за изтриване, е истина и след изпълнение на действието.
- Независими подцели: цели, които могат да се достигнат независимо една от друга в произволен ред. Пример: начално състояние = $\{on(a,table), on(b,table), on(c,table), on(d,table), clear(a), clear(b), clear(c), clear(d)\}$, цел = $\{on(a,b), on(c,d)\}$.
- Зависими (взаимодестващи си) цели: зависимост от реда на изпълнение на целите. Пример: начално състояние = $\{on(a,table), on(b,table), on(c,table), clear(a), clear(b), clear(c)\}$, цел = $\{on(a,b), on(b,c)\}$. Ако първо се изпълни $puton(b,c)$, а след това $puton(a,b)$ целта е удовлетворена. Какво става при обратен ред?
- Аномалия на Сузман (взаимно разваляща се цели): начално състояние = $\{on(a,table), on(b,table), on(c,a), clear(b), clear(c)\}$, цел = $\{on(a,b), on(b,c)\}$.
- Основен алгоритъм на STRIPS (+ и − са операции върху множества):

```
strips_plan(State,Goals,NewState,Plan) :-
%   [Goal|Rest_of_unsatisfied_goals] = Goals - State
    planning_rule(Action,Precondition,Add_list,Delete_list),
    member(Goal,Add_list),
    strips_plan(State,Precondition,State1,Plan1),
%   State2 = State1 + Add_list - Delete_list
    strips_plan(State2,Rest_of_unsatisfied_goals,NewState,Plan2),
%   Plan = Plan1 + [Action|Plan2]
strips_plan(State,_,State,[]).
```

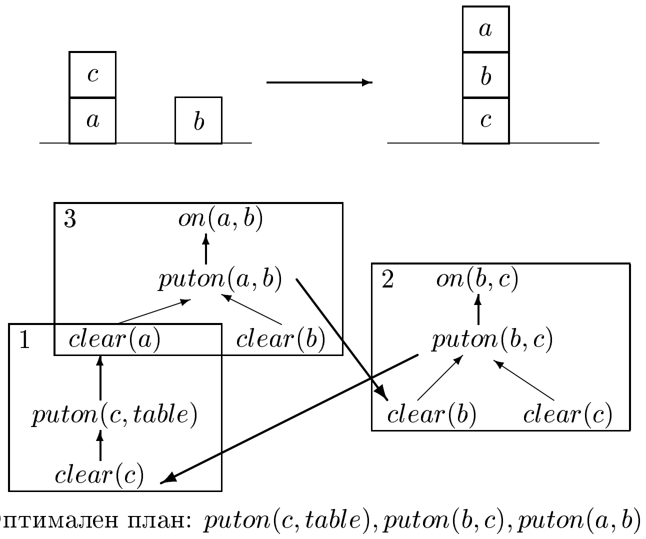


Figure 10.2: Планиране чрез наредба на частични планове

- Удовлетворяване на зависими цели: многократно изпълнение на основния алгоритъм.
- Оптималност на плановете (минимален брой действия) и ефективност на планиращите системи:
  - Пренареждане на цели.
  - Избор на правило с `Add_list`, включващ следващи цели (начално състояние = $\{on(a, table), on(b, table), on(c, a), clear(b), clear(c)\}$, цел = $\{on(a, c), on(c, b)\}$).
  - Подход на частично-наредени планове: търсене в пространсвото от частични планове (фигура 10.2):
    * план1 < план3 (действието на 1 осигурява предусловието на 3).
    * план1 < план2 (действието на 2 разваля предусловието на 1).
    * план2 < план3 (действието на 3 разваля предусловието на 2).
    * Следователно: план = {план1, план2, план3}.

10.2 Семинарни упражнения

- Алгоритъм за дедуктивно планиране на Пролог.
- Алгоритъм STRIPS на Пролог.
- Въвеждане на ограничения в "света на кубчетата" (напр. размер на кубчето и ограничение за поставяне на по-голямо кубче върху по-малко).
- Формализация на задачата за ханойските кули като задача за планиране.

10.3 Лабораторни упражнения

- Експерименти с дедуктивния планировчик (scalc.pl). Решаване на задачи с 4 и повече кубчета. Разширение на програмата за интерактивна работа (задаване на началното състояние от въпроса) и реализация на алгоритъм за итеративно търсене по нива.
- Сравнителни експерименти с дедуктивния планировчик (scalc.pl) и STRIPS (planner.pl). Сравнение по бързодействие и оптималност на плановете.