**планиране**.-**генерирането на последователност от действия за изпълнение на**

**задача или за постигане на определена цел,**

**Планирането декомпозира света до набор от логически условия и представя едно състояние**

**като конюнкция на положителни литерали.**

**Целта е достигната, ако състоянието съдържа всички литерали на целта или казано по друг**

**начин - състоянието s удовлетворява целта g,**

**Действие = Предусловие + Ефект**

**Първо, ще казваме, че действието е**

**приложимо**

**във всяко състояние, което отговаря на**

**предусловието.**

**10.3 Планиране с търсене в пространството от състояния**

Сега ще насочим вниманието си към планирането на алгоритми. Най-разбираемият подход е

като използваме търсене в пространството от състояния. Тъй като описанията на действията

определят както предусловията, така и ефектите, е възможно търсенето да се осъществи в

една от двете посоки:

Планиране чрез прогресия - търсене напред от началното състояние. Разглежда ефекта от

всички възможни действия в дадено състояние. – *фиг. (a)*

Планиране чрез регресия - търсене назад от целта. За достигане на целта, това което трябва да

бъде вярно е предишното състояние. – *фиг. (b)*

**10.4 Частично наредени планове**

Можем да изградим пълно нареден план, работейки с частично наредени

Планът, който изграждаме, включва две

независими последователности от действия - за ляв и десен крак. За разлика от пълните

наредби, тук не заявяваме строг ред на изпълнение на тези последователности.

Всеки алгоритъм за планиране**, в който могат да се поставят две действия в план,** без да се

уточнява кое се случва първо, се нарича

алгоритъм за частично наредени планове.

*Човек взаимодейства дистанционно с двама събеседници —*

*компютър и човек; на основание на отговорите на въпросите, тестовият субект е длъжен да определи с кого*

*разговаря — с машина (изкуствен интелект/компютърна програма) или с жив човек; задачата на*

*компютърната програма е да въведе човека в заблуда, карайки го по този начин да направи грешен избор*.

Едно от предимствата на логическите системи като формализъм за ПИЗ е, че преводът от

естествен към формален език е сравнително лесен, заради ясната им семантика и

изразителност. Има цели дялове в математиката, посветени на логиката, което я прави добър

избор за представяне на знания в машинен вид. Освен това програмните езици сами по себе

си са произлезли от логиката. В частност има програмни езици (като например Prolog), които

използват предикатната логика, което значително улеснява представянето и използването на

знания.

Недостатък на този формализъм е, че възникват проблеми при използване на неразрешими

или полуразрешими логики.

Има много варианти на семантични мрежи, но в същината си всички те представят знанията

чрез обекти, категории обекти и връзки между обектите.

Semantic Nets

– Adv.

Easy to follow hierarchy, easy to trace association, flexible

– Disadv.

Meaning attached to nodes might be ambiguous

exception handling is difficult

difficult to program

Характерно за този формализъм е, че знанието се представя чрез

двойки от вида: <условие;следствие>. Тези двойки образуват множество от правила,

определящи действието на системата. За всяка такава двойка агентът проверява дали

условието е издържано и ако е така, продукционното правило задейства следствието и то се

изпълнява.

**Comparisons of KR Methods**

Rules

– Adv.

simple syntax, easy to understand, simple interpreter, high

modular, flexible

– Disadv.

Hard to follow hierarchies, inefficient for large

Основна характеристика на фреймовете е, че описват знания, които винаги са верни и

структурата на тези знания и връзките между отделните такива е заложена в самия фрейм.

Слотовете във фреймовете могат да съдържат следната информация: информация за избора

на фрейм в дадена ситуация; информация за връзките на този фрейм с останалите;

процедури, които да се изпълнят след като определени слотове са запълнени; информация

по подразбиране, която да се използва в случай на липсваща (непопълнена) информация;

други фреймове (така се реализира йерархичната структура).

**Comparisons of KR Methods**

Frames

– Adv.

Expressive power, easy to set up slots for new properties

and relations

easy to create specialized procedures

easy to include default information and detect missing

values

– Disadv.

Difficult to program

difficult for inference

Four basic types in order of increasing generality:

– simple reflex agents

– reflex agents with state

– goal-based agents

– utility-based agents

All these can be turned into learning agents