Изкуствен интелект - летен семестър, 2023/2024 учебна година

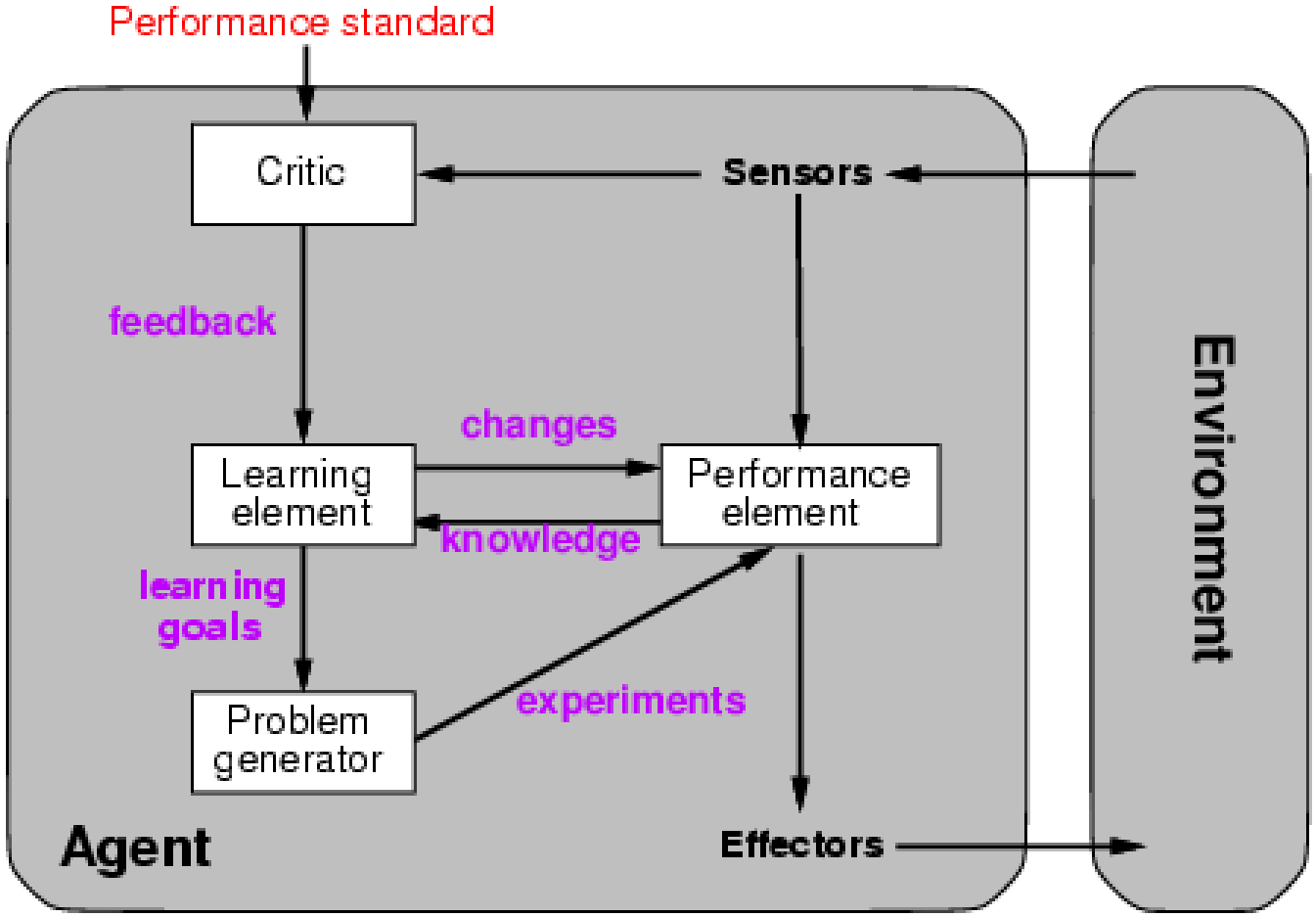
Тема 10:
Машинно самообучение

Същност на машинното самообучение

Х. Саймън: „Самообучението означава настъпване на такива промени в системата, които са адаптивни в смисъл, че позволяват на тази система при всеки следващ опит да извършва дадена работа по-ефективно и по-ефектно, отколкото при предишните опити“.

Р. Михалски: „Самообучението е конструиране или модифициране на различни представяния на предмета на дейност“.

Самообучаващ се агент:



Learning agent = performance element + learning element

Типове машинно самообучение в зависимост от характера на обратната връзка:

- Машинно самообучение (МС) с учител (ръководено МС, supervised machine learning)
- Машинно самообучение без учител (неръководено МС, unsupervised machine learning)
- Подсилено самообучение – когато обект с ИИ функционира в среда, в която обратната връзка се получава с известно закъснение

Самообучение чрез наизустяване

Представлява най-примитивен тип машинно самообучение (МС), аналог на ученето наизуст (зазубрянето) при човешкото обучение.

Идея. Запазват се резултати от работата на системата (условията на тежки, трудоемки задачи и получените решения) с цел използването им наготово в случай, че постъпи заявка за решаване на вече позната задача. В чистия си вид методът не предполага никаква допълнителна обработка на тези резултати. Оказва се, че при редица задачи такова запазване води до чувствително подобряване на работата на системата (повишаване на бързодействието, икономия на памет и др.).

Самообучение чрез макрооператори

Идея. Същата, както при самообучението чрез наизустяване – вариант на този тип МС, при който се предполага допълнителна обработка (обобщаване) на съхраняваните резултати от работата на системата. Използва се най-често в системи, свързани с планиране на действията.

Пример. Системата STRIPS има обучаваща подсистема за натрупване на резултати от вече извършена от нея работа под формата на макрооператори. Всеки макрооператор представлява обобщен план за решаване на даден тип задачи и се състои от *предусловия* (обобщават съществена част от началното състояние при решаването на конкретна задача за планиране), *тяло* (обобщение на конструирания от системата план) и *следусловия* (обобщение на целта при решаването на конкретна задача за планиране).

Самообучение чрез запомняне

Идея. Подход за самообучение с учител, при който целта е на базата на множество зададени от учителя решени примери (т. нар. *обучаващи примери*) за обекти, принадлежащи на *множество от известни класове*, да се класифицира определен *тестов пример*.

Най-използваният метод за самообучение чрез запомняне е *методът на най-близкия съсед* (NN – Nearest Neighbour). При този метод класификацията на тестовия пример зависи от степента на неговото сходство с единствен пример на понятие (клас) – този, който се намира на най-малко разстояние от него (неговия най-близък съсед).

Приема се, че тестовият пример принадлежи на този клас (това понятие), чийто представител е най-близо до него.

Друг често използван в практиката метод за самообучение чрез запомняне е *методът на k най-близки съседи* (*k*-NN). Числото *k* определя броя на най-близките екземпляри на понятия (класове) измежду обучаващите примери, които участват в определянето на решението за класификация на тестовия пример. Тестовият пример се класифицира в съответствие с (като представител на) класа, който се среща най-често сред неговите *k* най-близки съседи. Ако повече от един клас се среща най-често сред най-близките *k* съседи на тестовия пример, той обикновено се класифицира в съответствие с класа на най-близкия свой съсед сред конкуриращите се класове (най-близките негови *k* съседи).

Самообучение чрез уточняване на параметрите

Идея. Много системи с изкуствен интелект използват оценяващи (или разделящи) функции, които представляват комбинации от стойности на подходящо подбрано множество от параметри (признаци, фактори), като всеки параметър участва в оценяващата функция със съответен коефициент (тегло). При проектирането и първоначалното програмиране на такива системи обикновено е трудно априори да се определят правилно теглата на всички фактори, които участват във формулировката на оценяващата функция. Един възможен подход за правилното определяне на теглата на отделните фактори е свързан с включването на средства за модифициране (доуточняване) на тези тегла на основата на натрупания опит от работата на системата с текущия вариант на оценяваща функция.

Пример: програма на Samuel за игра на шашки (МС чрез наизустяване + МС чрез уточняване на параметрите)

Това е един от най-често използваните методи за МС в областта на разпознаването на образи и при невронните мрежи.

Самообучение чрез съвети

Идея. Знанията се придобиват от учител (евентуално учебник или др.) и съответната обучаваща подсистема трансформира тези знания във вид, който е използваем от обработващата подсистема. С други думи, учителят избира съдържанието, структурата и представянето на знанията (съвета), които иска да добави към съществуващите знания на системата, а обучаващата подсистема има за задача синтактичното преобразуване (преформулировката) на знанията, получени от учителя, във вид, който е достъпен за съответния обработващ (използващ знанията) модул.

Пример 1. В шахмата съвет от рода на „Стремете се да контролирате централната част на дъската“ е практически неизползваем за съответната програма за игра на шах, ако не съществува обучаваща програма, която да използва този съвет, като на негова основа коригира съответната оценяваща позициите функция (например като добави нов фактор за отчитане на броя на централните полета, които се контролират, т.е. могат да се атакуват от фигурите на съответния играч).

Пример 2. Програмата TEIRESIAS, която играе ролята ня интерфейс за получаване на съвети към популярната експертна система MYCIN. TEIRESIAS извършва специализации и обобщения на базата на съвети, давани от учител, като основава работата си на две прости идеи: разчита на воден от меню диалог, който прави нейната вътрешна структура максимално ясна за учителя; когато е необходим вход на естествен език, програмата се консултира с учителя, за да се убеди, че е превела (разбрала) правилно входния текст.

Самообучение чрез примери

Идея. Това е тип МС (частен случай на т. нар. *индуктивно самообучение*), което обикновено се прилага при системи, предназначени за решаване на задачи, свързани с класификация на обекти.

Класификацията е процес, при който по даден обект (описанието на даден обект) се получава името на класа, към който принадлежи този обект. Класификацията е важен етап от решаването на много задачи от областта на ИИ. При създаването на системи за решаване на такива задачи преди всичко трябва да бъдат дефинирани класовете, с които ще се работи. Често е много трудно да се даде пълна и вярна дефиниция на отделните класове в разглежданата област. Поради това е полезно създаването на самообучаващи се програмни системи, които могат сами да изграждат дефинициите на класовете от предметната област. Задачата за изграждането на дефинициите на класове е известна като *изучаване на понятия* (concept learning). МС чрез примери е най-популярният метод за изучаване на понятия.

Характерно за този тип МС е използването на множество от т. нар. ***обучаващи примери***, които могат да бъдат от два типа: *положителни* и *отрицателни* обучаващи примери.

Ще разгледаме накратко два популярни подхода при МС чрез примери: подхода на т. нар. *отделяне* и подхода на т. нар. *покриване.*

Подход на т. нар. ***отделяне***: целта е *примерите за даден клас да се отделят* от тези за другите класове. Типичен представител на такава *разделяща стратегия* е подходът на *класификационните дървета*.

Построяване на класификационни дървета

Класификационно дърво (КД): Всеки *възел* в едно КД представя даден атрибут, срещащ се в примерите. *Дъгите*, излизащи от даден възел, представят различните възможни стойности на този атрибут. Всеки *лист* на дървото определя класификацията на даден пример, като стойностите на съответните атрибути са зададени по пътя от корена към този лист.

В случай на едно понятие листата се маркират с “+” (или T, True) за положителните примери и с “-“ (или F, False) – за отрицателните примери.

Примерна задача. Да се вземе решение дали да се чака за маса в ресторант на базата на конкретните стойности на следните атрибути, които показват:

- ✓ *Alternate*: дали наблизо има друг подходящ ресторант
- ✓ *Bar*: дали ресторантът разполага с удобно място за изчакване
- ✓ *Fri/Sat*: дали денят е петък или събота
- ✓ *Hungry*: дали сме гладни
- ✓ *Patrons*: колко хора има в ресторанта (възможни стойности: None, Some, Full)

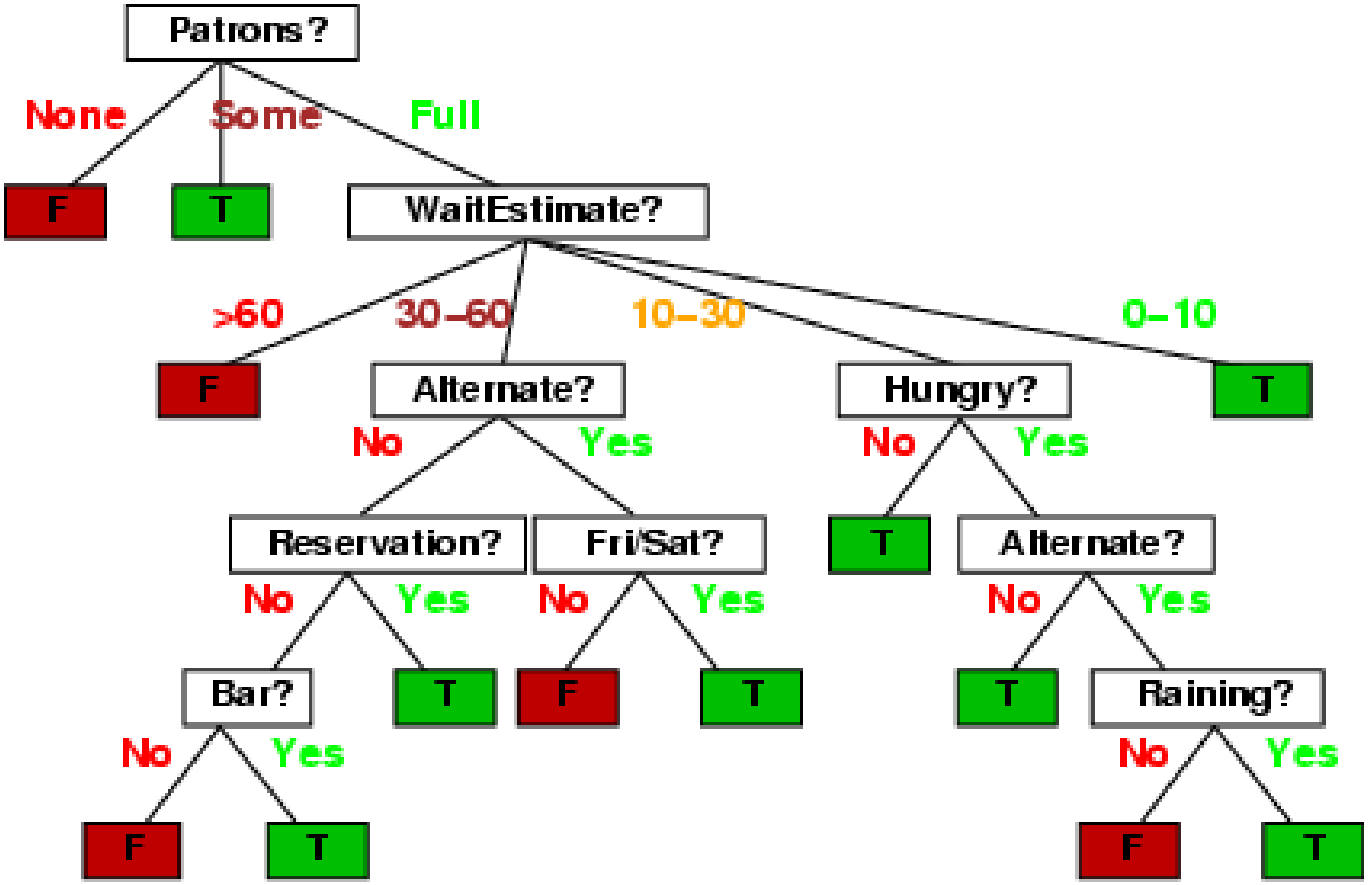
- ✓ *Price*: какво е нивото на цените в ресторанта ($, $$, $$$)
- ✓ *Raining*: дали навън вали
- ✓ *Reservation*: дали имаме предварително направена резервация
- ✓ *Type*: типът на ресторанта (френски, италиански и т.н.)
- ✓ *WaitEstimate*: предполагаемото време за чакане според персонала (0-10 минути, 10-30, 30-60, >60)

- Примерите се описват чрез подходящи стойности на атрибутите (булеви, дискретни, непрекъснати)
- Примерни ситуации, при които потребителят ще реши да чака (или да не чака) за маса:

Example	Attributes										Target
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*Wait*
X_1	T	F	F	T	Some	$$$	F	T	French	0–10	T
X_2	T	F	F	T	Full	$	F	F	Thai	30–60	F
X_3	F	T	F	F	Some	$	F	F	Burger	0–10	T
X_4	T	F	T	T	Full	$	F	F	Thai	10–30	T
X_5	T	F	T	F	Full	$$$	F	T	French	>60	F
X_6	F	T	F	T	Some	$$	T	T	Italian	0–10	T
X_7	F	T	F	F	None	$	T	F	Burger	0–10	F
X_8	F	F	F	T	Some	$$	T	T	Thai	0–10	T
X_9	F	T	T	F	Full	$	T	F	Burger	>60	F
X_{10}	T	T	T	T	Full	$$$	F	T	Italian	10–30	F
X_{11}	F	F	F	F	None	$	F	F	Thai	0–10	F
X_{12}	T	T	T	T	Full	$	F	F	Burger	30–60	T

- Примерите се разделят на положителни (+/T) и отрицателни (-/F)

Примерно класификационно дърво:



```
function DTL(examples, attributes, default) returns a decision tree
    if examples is empty then return default
    else if all examples have the same classification then return the classification
    else if attributes is empty then return MODE(examples)
    else
        best ← CHOOSE-ATTRIBUTE(attributes, examples)
        tree ← a new decision tree with root test best
        for each value v_i of best do
            examples_i ← {elements of examples with best = v_i}
            subtree ← DTL(examples_i, attributes − best, MODE(examples))
            add a branch to tree with label v_i and subtree subtree
        return tree
```

Забележка. Ако няма останали атрибути за избор, но класификацията на примерите все още не е приключила (има останали както положителни, така и отрицателни обучаващи примери), това означава, че съществува проблем. Останали са обучаващи примери, които имат еднакви стойности на атрибутите, но се класифицират по различни начини. Такава ситуация може да възникне, когато данните са некоректни (зашумени), когато избраните атрибути са недостатъчни или когато областта е недетерминирана.

В тези случаи функцията DTL връща резултат, който се определя от функцията MODE.

Естествено, видът на полученото класификационно дърво зависи от реда на ***избор на атрибути***.

При конструирането на класификационни дървета целта е да се построи *„компактно“ дърво с минимален брой разделящи възли* (т.е. целта е да се минимизира необходимият брой тестове). Така за предпочитане са плитките и силно разклонени дървета.

Постигането на тази цел се определя от избора на „добри“ атрибути. Критерий за качеството на даден атрибут е способността му да разделя множеството от примери на групи от еднородни примери. Формално това най-често се прави с използване на т. нар. *информационен критерий*.

Избор на „най-добрия“ атрибут

Основната идея на информационния критерий е да се изчисли *ентропията* на разпределението на обучаващите примери по класове, получено при разбиването на множеството от примери чрез стойностите на всеки атрибут. Най-добрият атрибут е този, при който се получава най-малка стойност на ентропията.

Оценката на ентропията (информационното съдържание) се основава на теорията на информацията на Шенън. Ентропията е обща оценка на нивото на „изненада“ или „неопределеност“ при получаването на n различни съобщения и се изчислява по формулата

$I = - \sum P_i \log_2 P_i$, където P_i ($i = 1, \ldots, n$) са вероятностите на отделните съобщения.

Предполага се, че всички примери от обучаващата извадка се срещат с една и съща вероятност. Тогава на базата на горната формула може да се оцени информационното съдържание (ентропията) $I(E)$ на даденото множество от обучаващи примери E. Същата оценка се отнася и за всяко класификационно дърво, което съответства на (покрива) това множество.

Ентропията може да се използва за оценка на качеството на избора на конкретен атрибут за разделящ възел в класификационното дърво. За тази цел се използва разликата между общото количество информация в дървото преди избора на атрибута и остатъка от информация, необходима за довършване на дървото с корен избрания атрибут.

Да предположим, че атрибутът *A* с *n* възможни стойности е избран за разделящ възел. Тогава той става корен на *n* поддървета, разделящи текущото множество от обучаващи примери *E* на *n* подмножества: E_1, E_2, ... , E_n. Може да се покаже, че *информацията, необходима за довършване на дървото след избора на A като корен*, е

$$R(A) = \sum_{i=1}^{n} \frac{|E_i|}{|E|} I(E_i)$$

Тогава информационната печалба от избора на атрибута A се пресмята по формулата

$$gain(A) = I(E) - R(A)$$

На всяка стъпка от работата на алгоритъма се избира атрибутът, който носи най-голяма информационна печалба.

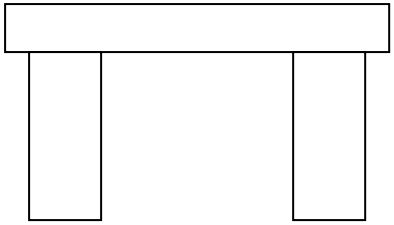
Подход на т. нар. ***покриване***: целта при построяване на описанията е те да *покриват примерите* (положителните примери) за дадения клас. По-точно, за всеки клас се задава (пълно) множество от положителни обучаващи примери (описания на обекти, принадлежащи на класа) и множество от отрицателни обучаващи примери (описания на обекти, които не принадлежат на класа, но приличат на обекти от положителните примери). Извършва се *обобщаване* на положителните примери (по такъв начин, че полученото описание на класа да включва като частни случаи описанията на всички положителни примери) и *ограничаване* на отрицателните примери (по такъв начин, че описанието на класа да изключва описанията на отрицателните примери).

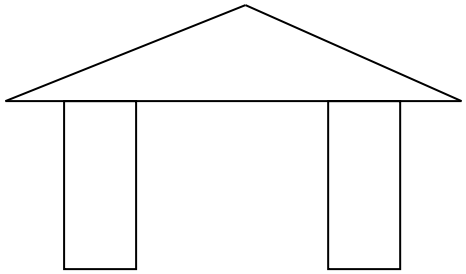
Пример: програмата ARCHES на P. Winston за класификация на обекти от света на кубчетата.

ARCHES използва структурно описание на дефинициите на понятията от света на кубчетата. Тя съдържа множество процедури за въвеждане на стилизирани рисунки на съответните обучаващи примери, които анализират фигурите от примерите и конструират подходящи семантични мрежи – структурни описания на различните обекти.

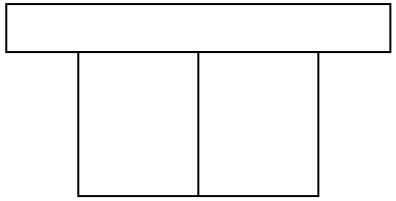
Обучаващи примери за понятието „арка“ (arch):

положителни примери



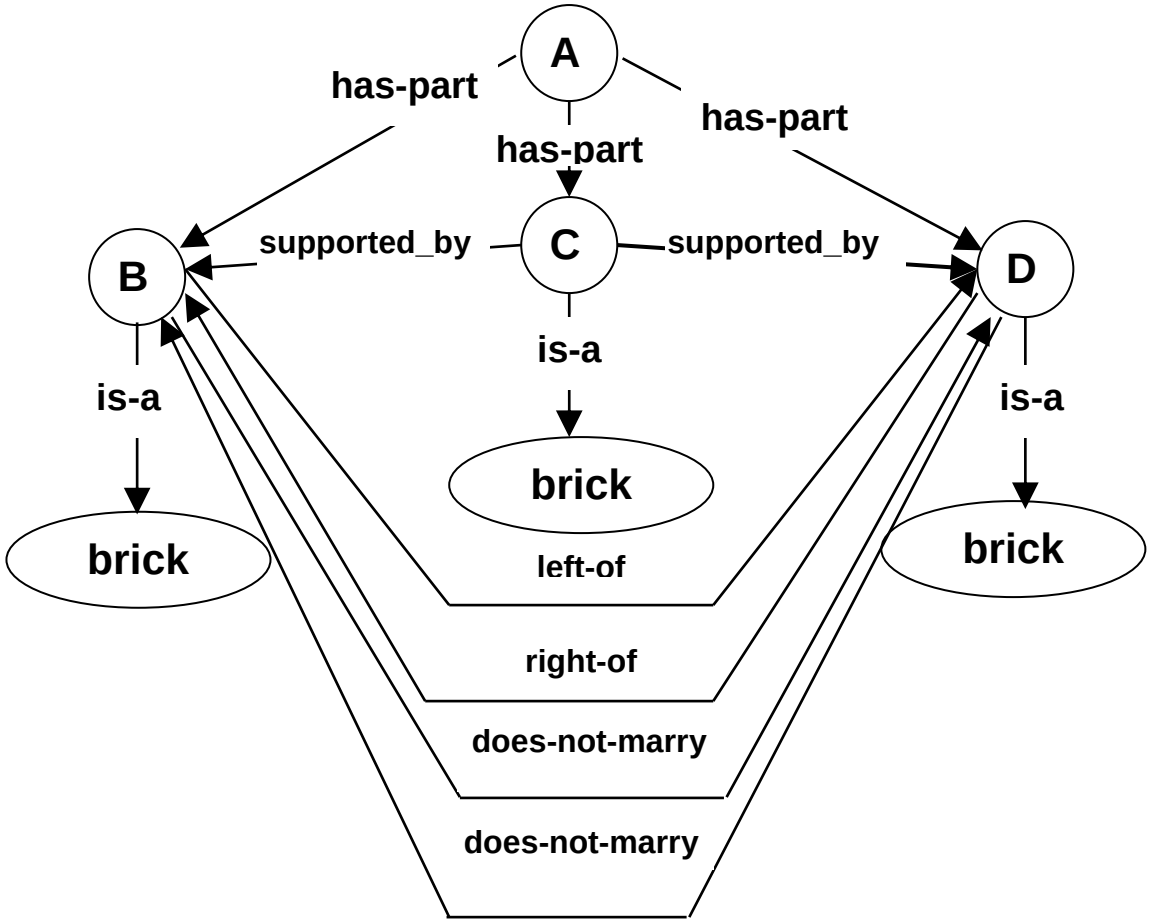


отрицателни примери

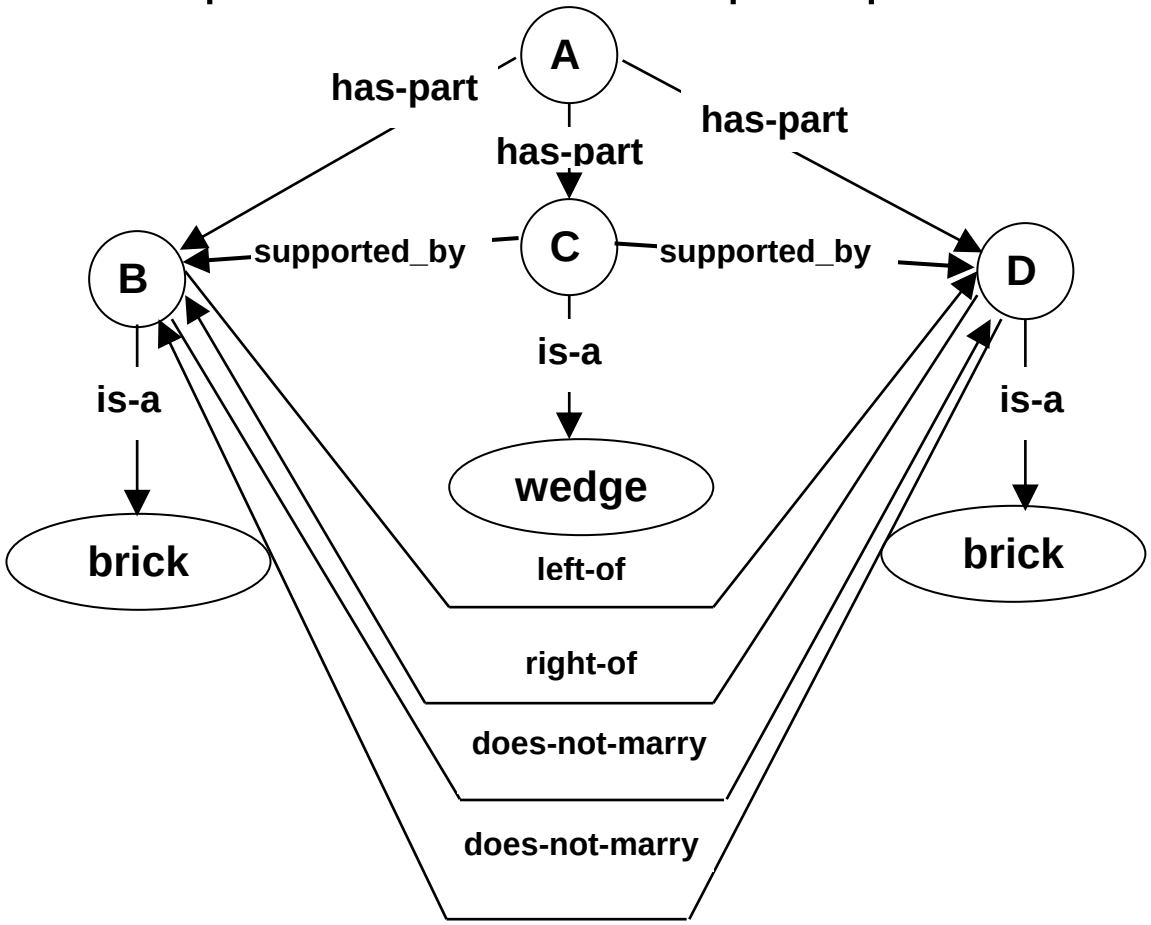


Създаване на описанието на изучаваното понятие (арка): чрез обобщение на описанията на положителните примери (което да покрива множеството на положителните обучаващи примери) и ограничаване на отрицателните обучаващи примери (така че обобщеното описание на положителните примери да изключва отрицателните). За целта се използват множество подходящи евристики.

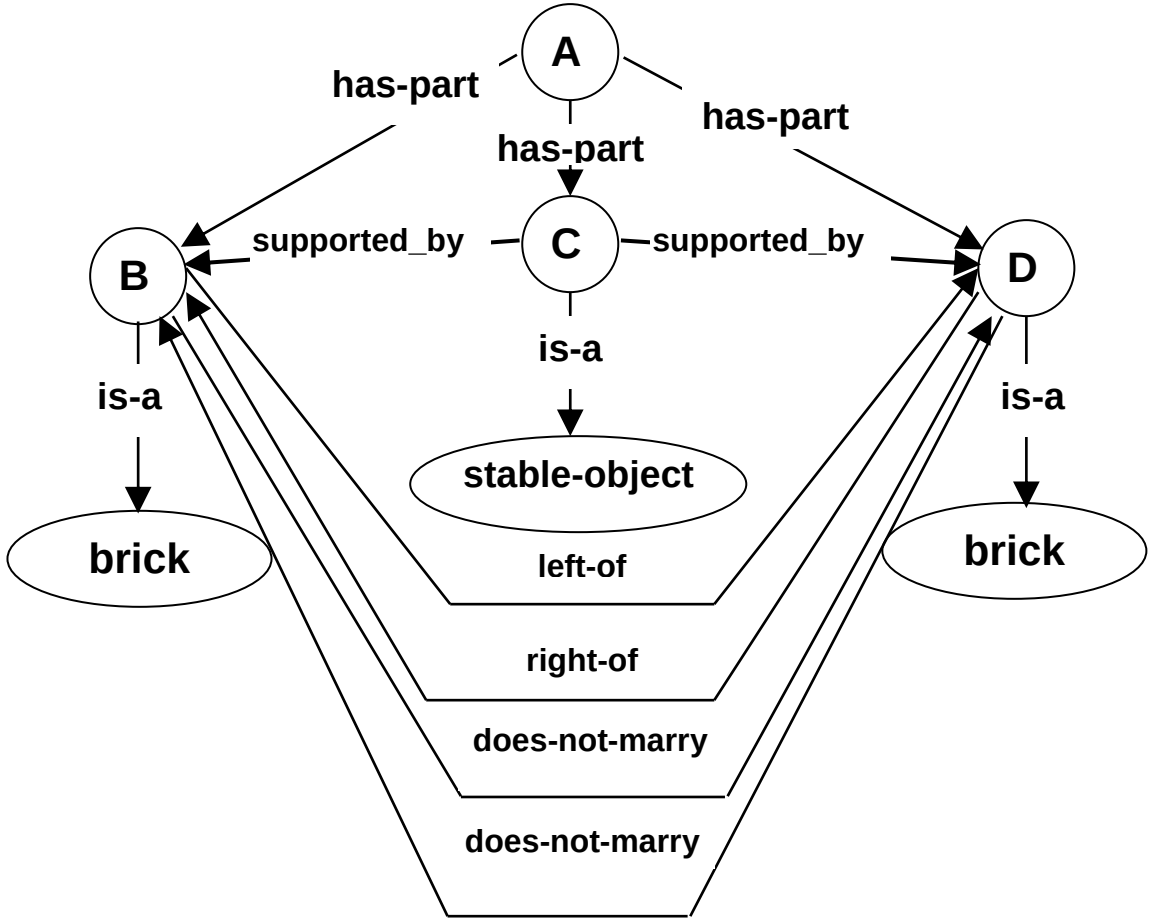
Описание на първия положителен пример:



Описание на втория положителен пример:

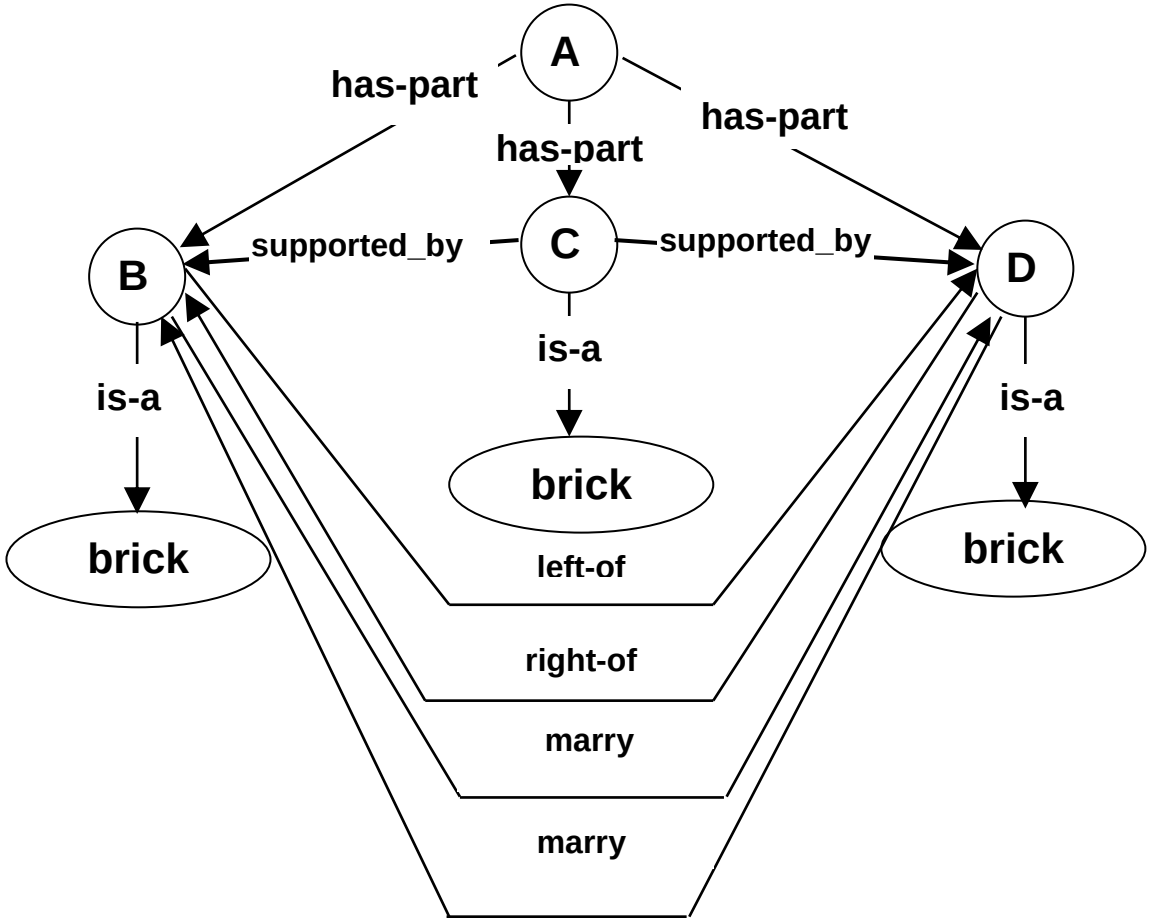


Обобщение на описанията на двата положителни примера:



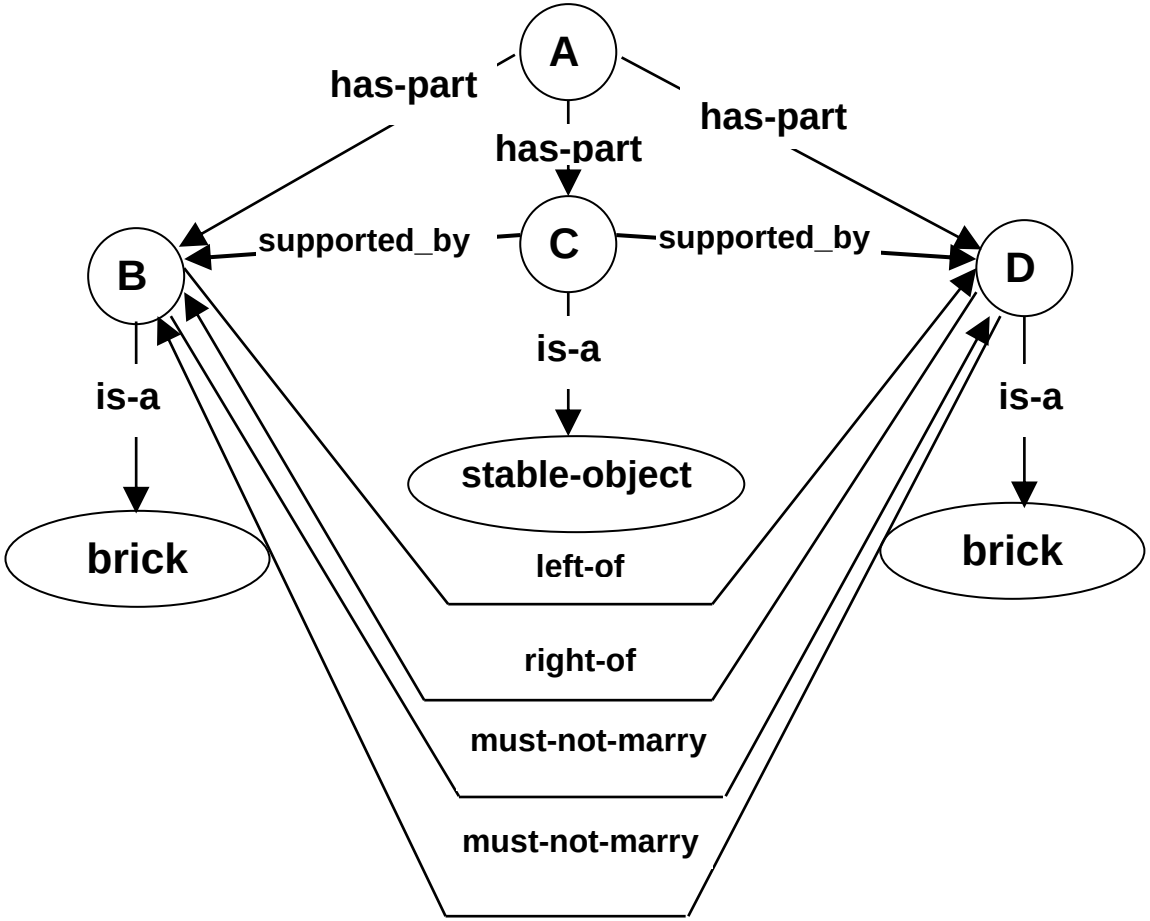
Забележка. Предполага се, че stable-object е най-близкият общ предшественик в йерархията на геометричните форми на формите brick и wedge. С други думи, обобщаването на положителните примери е свързано с минимално обобщаване на съответните геометрични форми.

Описание на отрицателния пример:



Огранчаване на отрицателния пример: чрез засилване на името на връзката “does-not-marry” до “must-not-marry” (за положителните примери е изпълнено “does-not-marry”, за отрицателния е изпълнено обратното, следователно валидната за положителните примери релация е решаваща за дефинирането на изучаваното понятие).

Окончателно описание на понятието „арка“:



Самообучение чрез обяснение

Идея. При МС чрез примери съществено е честото използване на голям брой обучаващи примери (положителни и отрицателни). Тук се използва един обучаващ пример и на основата на допълнителни знания за предметната област се конструира обяснение защо той е пример за разглежданото понятие. Чрез обобщение на обяснението се получава описанието на изучаваното понятие.

При самообучението чрез обяснение съответната обучаваща програма използва следните видове данни/знания:

- *Целево понятие* (описание на високо равнище на целевото понятие в термините на предметната област, но без да е изпълнен т. нар. критерий за операционалност);
- *Критерий за операционалност (конструктивност)* – описание или списък на понятията, в чиито термини трябва да бъде формирано описанието на целевото понятие;
- *Обучаващ пример* – описание на обект от предметната област, който е положителен пример за целевото понятие;
- *Теория за предметната област* – множество от правила, които описват необходимите знания за класификация на обектите в предметната област (в термините на критерия за операционалност).

На основата на тези данни и знания обучаващата програма конструира обяснение (доказателство например чрез използване на обратен извод върху теорията за предметната област) защо обучаващият пример е пример за изучаваното (целевото) понятие, което е формулирано в термините на критерия за операционалност. Чрез обобщение на това обяснение се получава търсеното описание на целевото понятие.

Пример (Митчел и др., 1986 г.): *обучение на понятието „чаша“*. Съгласно отбелязаното по-горе, съответната система разполага с:

1. *Обучаващ пример – обект23*

собственик(обект23, Джон) ^ Е(обект23, лек) ^
цвят(обект23, кафяв) ^ има-част(обект23, дръжка16) ^
има-част(обект23, дъно19) ^ има-част(обект23, вдлъбнатина12) ^
.....

Очевидно някои от характеристиките (признаците) на *обект23* са по-съществени за определянето му като чаша от други. Изолирането на действително важните признаци от гледна точка на описанието на съответното понятие при този метод става с помощта на знанията (теорията) за предметната област.

2. *Целево понятие – чаша*: предметът *x* е чаша тогава и само тогава, когато има свойствата *премествам*, *отворен* и *стабилен-съд*.

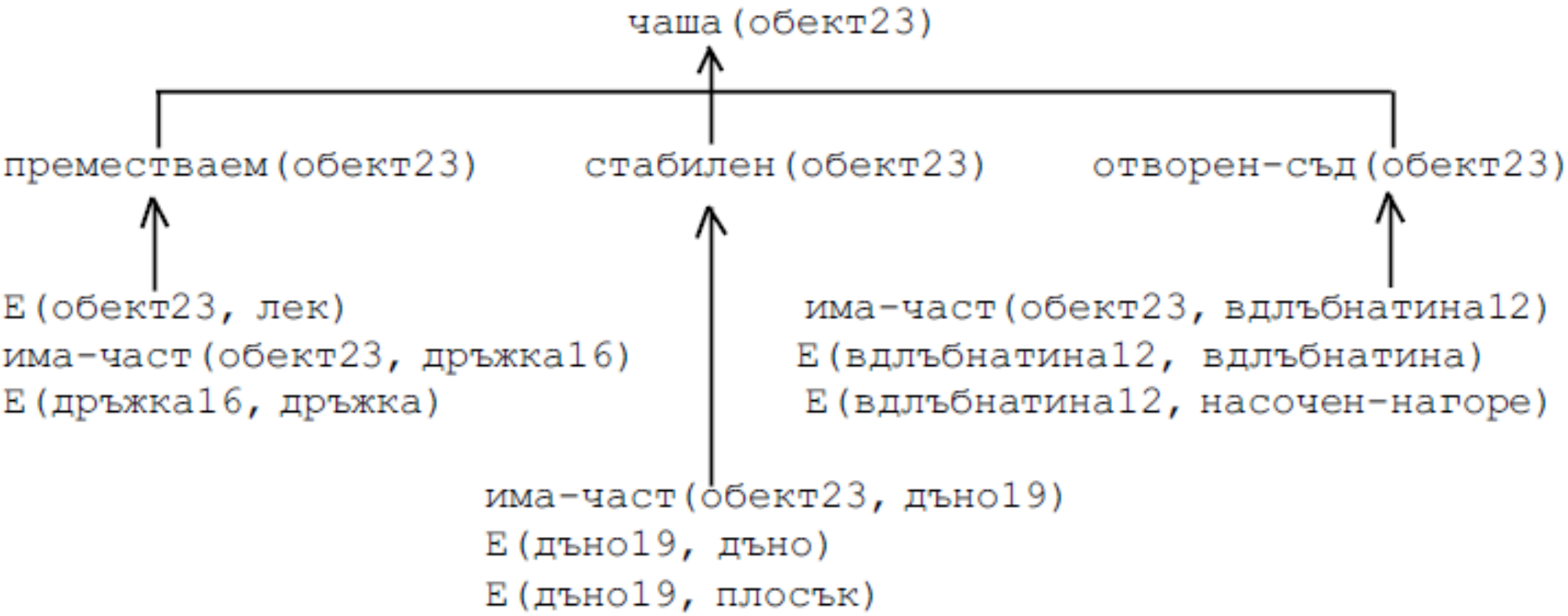
3. *Критерий за конструктивност (операционалност)*: описанието на понятието трябва да бъде изразено в прости структурни термини (*лек*, *плосък*, *има-част* и т.н.).

4. Теория (знания) за предметната област

…..
Е(х, лек) ^ има-част(х, у) ^ Е(у, дръжка) ⇒ премествӑем(х)
има-част(х, у) ^ Е(у, дъно) ^ Е(у, плосък) ⇒ стабилен(х)
има-част(х, у) ^ Е(у, вдлъбнатина) ^ Е(у, насочен-нагоре) ⇒
отворен-съд(х)
…..

Задачата, която решава обучаващата система, е формиране на конструктивно описание на понятието *чаша*.

Първата стъпка е построяване на обяснение на това, защо *обект23* е чаша. Това може да стане чрез построяване на доказателство, както е показано на следващата фигура. За целта могат да се използват стандартните методи за извод при продукционните системи.


чаша(обект23)
премества(обект23)
стабилен(обект23)
отворен-съд(обект23)
Е(обект23, лек)
има-част(обект23, дръжка16)
Е(дръжка16, дръжка)
има-част(обект23, дъно19)
Е(дъно19, дъно)
Е(дъно19, плосък)
има-част(обект23, вдлъбнатина12)
Е(вдлъбнатина12, вдлъбнатина)
Е(вдлъбнатина12, насочен-нагоре)

Горното доказателство изолира съществените признаци на обекта, представен от обучаващия пример. То е и основа за извършване на ***обобщение***. Ако обединим всички използвани предположения и заменим константите с променливи, получаваме следното описание на понятието *чаша*:

има-част(x, y) ^ E(y, вдлъбнатина) ^ E(y, насочен-нагоре) ^
има-част(x, z) ^ E(z, дъно) ^ E(z, плосък) ^
има-част(x, w) ^ E(w, дръжка) ^ E(x, лек)

Това описание удовлетворява критерия за конструктивност и може да бъде използвано например от робот за класификация на обекти от предметната област.

Най-съществено за обучението чрез обяснение е използването на знания за предметната област. Тези знания по принцип са достатъчни за извършването на класификация на обектите от областта. Използването на обучаващ пример за разглежданото понятие дава възможност за по-голяма ефективност и конструктивност. Примерът насочва и ограничава частта от знанията за областта, които са приложими в конкретния случай. В противен случай в процеса на извод ще се използва цялата БЗ за предметната област.