»Лекционен курс

»Интелигентни системи

Предикатна логика

Virtual
learning space

Увод

» В предишните лекции показахме как един агент, използващ знания, може да представя света, в който оперира, и да прави заключения за действията, които да предприеме

» Използвахме съждителна логика като език за представяне

> Достатъчна за да илюстрира основните понятия на логиката и агентите, базирани на знания

» За съжаление, СЛ е твърде слаба за представяне знания за сложни среди по един ясен начин

» Ще разглеждаме предикатната логика - достатъчно изразителна за представяне по походящ начин знанията от общ характер

Предикатна логика

» Езикът на предикатната логика е изграден около понятията за обекти и релации

» То е толкова важно за математиката, философията и изкуствения интелект именно защото тези области (всъщност голяма част от ежедневното човешко съществуване) могат да бъдат разглеждани като работа с обекти и релации (отношения) между тях

» Предикатната логика може също да изразява факти за някои или всички обекти

  > Това дава възможност да се представят общите закони или правила

  > Напр., “Полетата, съседни на златото, блестят"

Предикатна логика

- СЛ изхожда от това, че фактите важат или не важат в света – всеки факт може да се намира в едно от двете състояния: вярно или грешно
  - И всеки модел присвоява на всеки съждителен символ true или false
- ПЛ предоставя повече
  - Светът се състои от обекти, които стоят в определени отношения помежду си, които важат или не важат
  - Съответно формалните модели са по-сложни от тези на СЛ

Модели за ПЛ

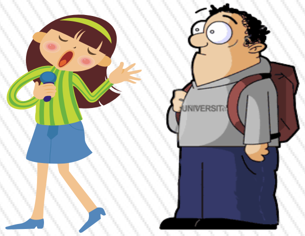
- Модел: формална структура, която образува възможен свят
  - Всеки модел свързва лексиката на логическите изрази с елементи на възможния свят, така че да може да бъде установена истинността на всеки израз
  - Модели за СЛ – свързват съждителни символи с предварително дефинирани верностни стойности
- Модели за ПЛ: значително по-интересни
  - Съдържат обекти – домейн на един модел е множеството на обектите, които той съдържа
  - Не трябва да бъде празно (поне един елемент)
  - Няма ограничение за обектите
- Строго разгледано: моделите в ПЛ предполагат тотални функции

Модели в ПЛ

- От математическа гледна точка няма значение какви са обектите – важното е колко са в един модел
- За примера - предполагаме, че имаме следните 5 обекта:
  - Марийка, Иванчо, раница, ляв крак на Марийка и ляв крак на Иванчо

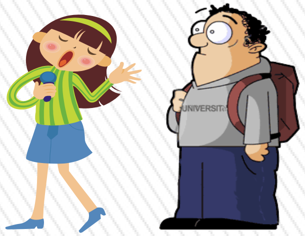
- Обектите в модела се намират в различни отношения
  - Напр., Марийка и Иванчо са приятели
- Формално: една релация е множество от подредени n-торки (<…>)
- Напр., релацията „приятели“ = {<Марийка, Иванчо>, <Иванчо, Марийка>}
- Релацията „на гърба“ = <раница, Иванчо> – има само една n-торка
- Двете релации са бинарни – свързват двойки обекти

Модели в ПЛ



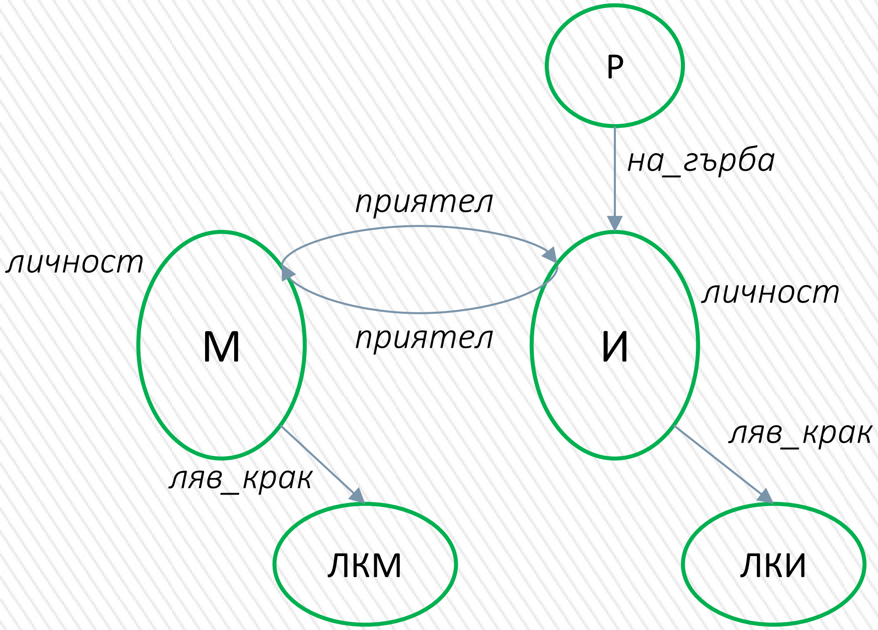
- Съществуват също унарни релации (свойства), като напр.:
  - „личност“ – вярно за Марийка и Иванчо
  - „девойка“ – вярно само за Марийка
  - „раница“ – вярно само за раница
- Някои отношения се представят най-добре като функции – един обект трябва да съответства точно на един обект, като напр.:
  - <Марийка> ⟶ ляв крак Марийка
  - <Иванчо> ⟶ ляв крак Иванчо

Модели в ПЛ



- Строго разгледано моделите в ПЛ предполагат тотални функции, т.е. за всяка входна n-торка съществува една стойност
  - Така раница трябва да има ляв крак, също ляв крак има ляв крак
  - Съществува техническо решение за този неприятен проблем, където се въвежда допълнителен „невидим“ обект, който представя левия крак на всички, които нямат ляв крак, включително на себе си
- Към съществените аспекти на един модел принадлежи свързването между тези елементи и лексиката на логическите изрази

Пример за модел



Синтаксис на ПЛ

- Основополагащи синтактични елементи са символите, които означават обекти, релации и функции
- Три вида символи: започват с големи букви
  - Константни символи – за обекти
  - Предикатни символи – за релации
  - Функционални символи – за функции
- Всеки предикатен и функционален символ има размерност (брой аргументи)

Синтаксис на ПЛ (BNF)

Sentence	$\rightarrow$	*AtomicSentence* | *ComplexSentence*
AtomicSentence	$\rightarrow$	*Predicate* | *Predicate*(*Term*, ...) | *Term* = *Term*
ComplexSentence	$\rightarrow$	(*Sentence*) | [*Sentence*]
	|	$\neg$ *Sentence*
	|	*Sentence* $\land$ *Sentence*
	|	*Sentence* $\lor$ *Sentence*
	|	*Sentence* $\Rightarrow$ *Sentence*
	|	*Sentence* $\Leftrightarrow$ *Sentence*
	|	*Quantifier* *Variable*, ... *Sentence*
Term	$\rightarrow$	*Function*(*Term*, ...)
	|	*Constant*
	|	*Variable*
Quantifier	$\rightarrow$	$\forall$ | $\exists$
Constant	$\rightarrow$	A | X_1 | *John* | $\cdots$
Variable	$\rightarrow$	a | x | s | $\cdots$
Predicate	$\rightarrow$	*True* | *False* | *After* | *Loves* | *Raining* | $\cdots$
Function	$\rightarrow$	*Mother* | *LeftLeg* | $\cdots$
OPERATOR PRECEDENCE	:	$\neg, =, \land, \lor, \Rightarrow, \Leftrightarrow$

Символи и интерпретации

- Както в СЛ, всеки модел трябва да доставя необходимата информация за това, дали едно съждение е вярно или грешно
  - За тази цел, всеки модел съдържа една интерпретация, която специфицира точно обектите, релациите и функциите, които се реферират от константни, предикатни и функционални символи

Пример за интерпретация

- *M* реферира обекта Марийка, *I* – съответно Иванчо
- *Friends* реферира релацията „приятели“, която е множеството от двойките обекти, принадлежащи към нея
- *OnBack* реферира релацията „на гърба“ между обектите „раница“ и „Иванчо“
- *Person*, *Backpack*, *Girl* реферират унарните релации „личност“, „раница“ и „девойка“
- *LeftLeg* реферира функцията за „ляв крак“

Символи и интерпретации

- Съществуват още много възможни интерпретации за тези символи за разгледания модел
  - Напр., в друга интерпретация M може да реферира „раница“ и I да реферира „левия крак на Марийка“
- Така за 5 обекта и 2 константни символа съществуват 25 възможни интерпретации
- Не всички обекти се нуждаят от име
  - Напр., преднамерената интерпретация може да не предвижда име за раницата или краката
- Също така е възможно даден обект да има няколко имена

Терми

- Терм: логически израз, отнасящ се към един обект
  - Константните символи са терми
- Не винаги е практично за обозначаване на един обект да използване отделен символ
  - Напр., левият крак на Мария
  - Точно за това са предвидени функционални символи
  - Вместо константен символ използваме LeftLeg(M)

Терми

- Комплексните терми се образуват от функционални символи, последвани от в скоби зададен списък на терми, изпълняващи ролята на аргументи
- Формална семантика на един терм $f(t_1, \ldots, t_n)$:
  - Функционалният символ f реферира някаква функция в модела (нека да бъде F)
  - Термите-аргументи реферират обекти от домейна (нека да са $d_1, \ldots, d_n$)
  - Като цяло един терм реферира обект
  - Напр., LeftLeg(M)
- По този начин интерпретацията фиксира референцията за всеки терм

Атомарни съждения

» След като имаме терми (реферират обекти) и предикатни символи (реферират релации) можем да ги комбинираме за да конструираме атомарни съждения

» Атомарните съждения (кратко атоми) изразяват факти

  > Пример: Брат(Георги, Иван)

» Атомите могат да имат комплексни терми като аргументи

  > Пример: Женен(Баща(Георги), Майка(Иван))

» Един атом е true в един модел, ако релацията, реферирана чрез предикатния символ е валидна за обекти, реферирани от аргументите

Комплексни съждения

- С помощта на познатите ни логически оператори можем да конструираме комплексни съждения
- Пример: Брат(Георги, Иван) ∧ Брат(Иван, Георги)

Квантори

- Логика, която работи с обекти
  - Естествено е да искаме да изразяваме свойства на цели колекции обекти
  - Вместо да изреждаме обектите по имена
- За целта можем да използваме квантори
- ПЛ дефинира два стандартни квантора:
  - Универсален ($\forall$)
  - Екзистенциален ($\exists$)

Универсален квантор

- Известна трудност при изразяване общи правила в СЛ
  - Напр., ∀: „за всички“ x: променлива (пишат се с малка буква)
- Променливите са терми и като такива могат да бъдат аргументи на функции
- Базов терм: терм, който не съдържа променлива
- Съждението ∀x P, където P произволен логически израз
  - P верен за всеки обект x

Екзистенциален квантор

- Съждение за един определен обект, без да използваме име
  - Пример: Интуитивно, ∃x P е вярно за най-малко един обект
- По-формално, ∃x P е вярно в един модел в една определена интерпретация, когато P е вярно в най-малко една разширена интерпретация, където x реферира един елемент от домейна

Вградени квантори

- Комплексни съждения с повече квантори
- В най-простия случай всички квантори са от един и същ тип
  - Пример: $\forall x \;\; \forall y \;\; Brother(x,y) \Rightarrow Sibling(x,y)\,.$
- Съществуват смесени случаи
  - Тук последователността на кванторите е съществена
  - Примери:
    - „Всеки обича някой“ - ∀x ∃y Обича(x,y)
    - „Има някой, обичан от всички“ - ∃y ∀x Обича(x,y)
- Могат да се използват скоби
  - Пример: ∀x (∃y Обича(x,y))

Връзка между кванторите

- Двата квантора са тясно свързани чрез отрицание
  - Пример1:
    - „Никой не обича спанак“: ∀x ¬ Обича(x,Спанак)
    - Еквивалентно на: ¬∃x Обича(x,Спанак)
  - Пример2:
    - „Всеки обича сладолед“: ∀x Обича(x,Сладолед)
    - Еквивалентно на: ¬∃x ¬ Обича(x,Сладолед)
- Понеже
  - ∀: всъщност конюнкция върху универсума на обекти
  - ∃: всъщност дизонюнкция върху универсума на обекти
  - Удовлетворяват Де Морган

Еквивалентност

- В ПЛ има също възможност за създаване на атоми
- Използваме символа за равенство „=“
  - Пример: Баща(Иван) = Георги
- Обектът, рефериран от Баща(Иван) и обектът, рефериран от Георги са един и същ

ПЛ за света на W.

» Някои аксиоми на СЛ за света на W. са дадени в предишните лекции

» Аксиомите на ПЛ са много по-кратки, представяйки по естествен начин точно това, което искаме да кажем

» Да си припомним, че W. агентът получава вектор от възприятия с пет елемента

» Кореспондиращото съждение в ПЛ, съхранено в БЗ, трябва да включва както възприятието, така и времето, в което е възникнало - в противен случай агентът ще се обърка за това, кога какво е видял

Вход и изход на агента

Възприятия на агента: типично съждение, където времената като цели числа

Бинарен предикат

$$\boxed{Percept([Stench, Breeze, Glitter, None, None], 5)}$$

Константи в списък *Константа*

Действия на агента

$$\boxed{Turn(Right),\quad Turn(Left),\quad Forward,\quad Shoot,\quad Grab,\quad Climb}$$

Терми

Вход и изход на агента

За да определи кое е най-доброто действие, програмата на агента изпълнява заявката и връща съответно резултата

$\text{ASKVARS}(\exists a \;\; BestAction(a, 5)) \longrightarrow \{a/Grab\}$

Сензорните данни за възприятията включват някои факти за текущото състояние

$$\forall t, s, g, m, c \;\; Percept([s, Breeze, g, m, c], t) \;\Rightarrow\; Breeze(t)$$
$$\forall t, s, b, m, c \;\; Percept([s, b, Glitter, m, c], t) \;\Rightarrow\; Glitter(t)$$

Вход и изход на агента

Простото „рефлексно” поведение може да се имплементира чрез квантифицирани импликации.

$$\boxed{\forall t \;\; Glitter(t) \;\Rightarrow\; BestAction(Grab, t)}$$

Средата на агента

- Нека започнем с обекти - очевидни кандидати са полета, ями, W.
- Бихме могли да назовем всеки квадрат - $Square_{1,2}$ и т.н. - но тогава фактът, че $Square_{1,2}$ и $Square_{1,3}$ са съседни, трябва да бъде „допълнителен“ факт и ще ни е необходим един такъв факт за всяка двойка полета.
- По-добре е да се използва сложен терм, в който ред и колона се показват като цели числа – напр., можем просто да използваме списъчния терм [1, 2]
- Съседството на всеки два квадрата може да бъде определено както следва:

$$\begin{aligned}&\forall\, x, y, a, b \;\; Adjacent([x, y], [a, b]) \;\Leftrightarrow \\ &\quad (x = a \wedge (y = b - 1 \vee y = b + 1)) \vee (y = b \wedge (x = a - 1 \vee x = a + 1))\end{aligned}$$

Средата на агента

- Можем да именуваме всяка яма, но това би било безсмислено, поради това, че няма причина да се прави разлика между отделните ями.
- По-лесно е да се използва единичен предикат Pit, който е валиден за полетата, съдържащи ями.
- Тъй като има точно един W.
- Една константа Wumpus е също толкова добра, колкото и един унарен предикат.

Средата на агента

- Местоположението на агента се променя с течение на времето, така че ще използваме At(Agent, s, t), за да означим, че агентът е в полето s в момент t
- Можем да фиксираме местоположението на W. с ∀t At(Wumpus, [2, 2], t).
- След това можем да кажем, че обектите могат да бъдат само на едно място в даден момент

$$\forall x, s_1, s_2, t \;\; At(x, s_1, t) \land At(x, s_2, t) \Rightarrow s_1 = s_2$$

Средата на агента

- Като се има предвид текущо местоположение, агентът може да изведе свойствата на полето от свойствата на актуалните му възприятия
- Напр., ако агентът е в поле и възприема полъх, може да заключи, че това поле е прохладно
- Полезно е да знаем, че полето е прохладно, защото знаем, че ямите не могат да се движат
- Освен това Breezy няма времеви аргументи

$$\boxed{\forall\, s, t \;\; At(Agent, s, t) \wedge Breeze(t) \;\Rightarrow\; Breezy(s)}$$

Средата на агента

- След като открихме кои места са прохладни (или миризливи) и, много важно, не прохладни (или не миришещи), агентът може да заключи къде са ямите (и къде е W.)
- Докато СЛ изисква отделна аксиома за всяко поле и ще се нуждаят от различен набор от аксиоми за всяко географско разположение на света, ПЛ просто се нуждае от една аксиома:

$$\boxed{\forall s \;\; Breezy(s) \;\Leftrightarrow\; \exists r \;\; Adjacent(r, s) \wedge Pit(r)}$$

Средата на агента

- По подобен начин в ПЛ можем да изразим квантифицираме в течение на времето, така че за всеки предикат се нуждаем само от една аксиома за състояния-наследници, а не от различно копие за всеки времеви етап
- Напр., аксиомата за стрелата е:

$$\forall t \;\; HaveArrow(t+1) \;\Leftrightarrow\; (HaveArrow(t) \wedge \neg Action(Shoot, t))$$

Твърдения и заявки в ПЛ

- Съжденията се добавят към БЗ, използвайки Tell
  - Както в СЛ
- Такива съждения наричаме твърдения
- Напр., можем да твърдим, че Борис е цар, Иван е личност и всички царе са личности:

Tell(KB, King(Boris)).
Tell(KB, Person(Ivan)).
Tell(KB, $\forall$x King(x) $\Rightarrow$ Person(x)).

Твърдения и заявки в ПЛ

- Можем да задаваме въпроси към БЗ, използвайки ASK
  - За примера:

Ask(KB, King(Boris)).

- Въпросите, зададени с ASK, се наричат заявки или цели
- Най-общо казано, на всяка заявка, която е логически изводима от БЗ, трябва да се отговори утвърдително (true)
- За примера: Ask(KB, Person(Boris)).

Твърдения и заявки в ПЛ

- Можем да правим квантифицирани заявки:

  Ask(KB, ∃x Person(x)).

- Отговорът е true, но това вероятно не е толкова полезно, колкото бихме искали
  - По-скоро е като да отговорим на „Може ли да ми кажете колко е часът?“ с „Да“

Твърдения и заявки в ПЛ

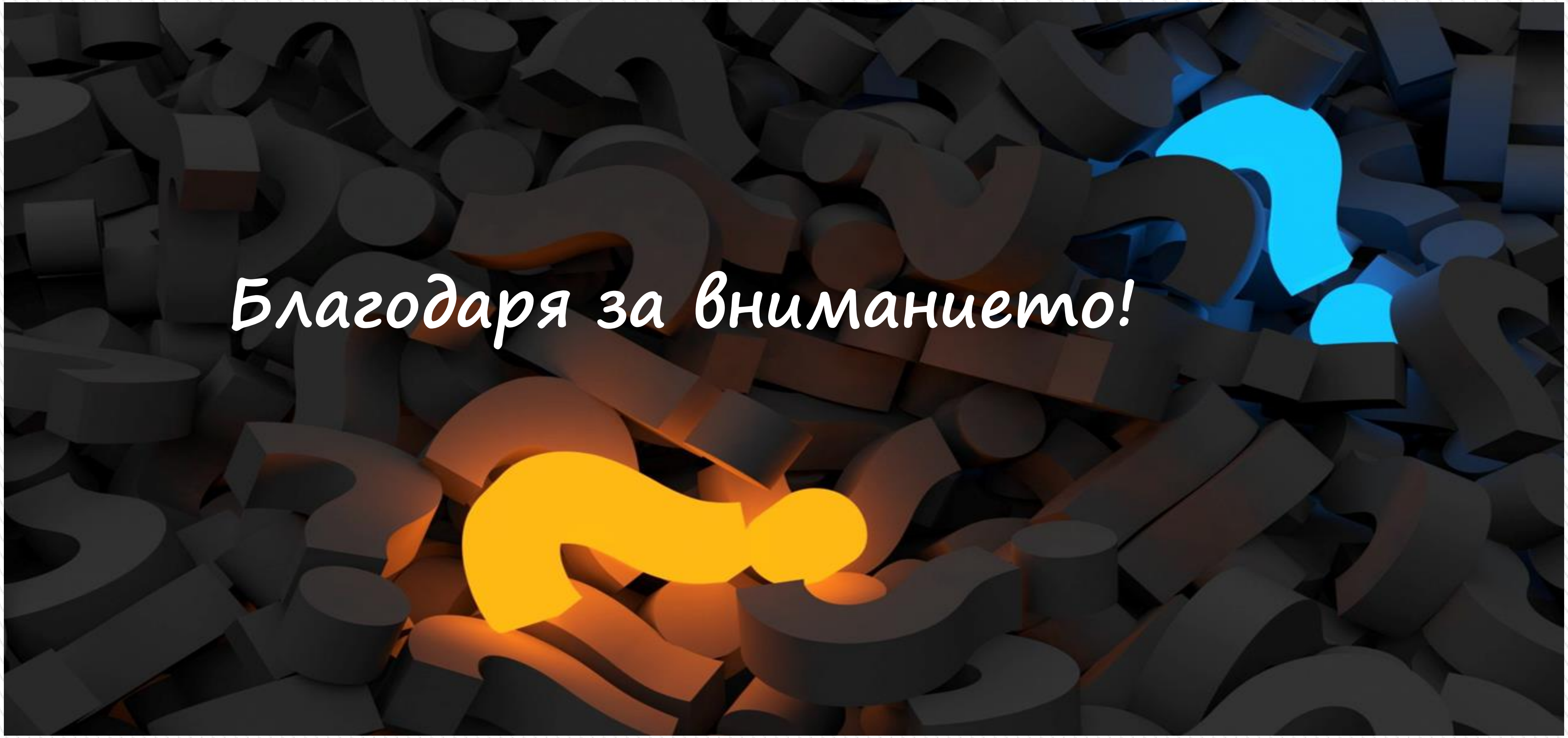
- Ако искаме да знаем каква стойност на x прави твърдението вярно, ще се нуждаем от различна функция:

AskVars(KB, Person(x)).

- Връща множество от отговори, в случая:

{x/Boris}, {x/Ivan}

- Такъв отговор се нарича субституция (или свързващ списък)


Благодаря за вниманието!