4-дәріс.

Білімдерді ұсынудың логикалық модельдері

Пәні: Интеллектуалды ақпараттық жүйелер

Өткен дәрісте біз интеллектуалды жүйеде білімді компьютерге қалай ұсынуға болатынын қарастырдық. Өнімдік ережелер, семантикалық желілер, фреймдер және онтологиялар сияқты білімді ұсыну тәсілдерімен таныстық.

Бүгін осы тақырыпты жалғастырып, білімді ұсынудың маңызды тәсілдерінің бірі — **логикалық модельдерді** қарастырамыз.

Алдымен қарапайым жағдайды қарастырайық.

Университеттің интеллектуалды кеңесші жүйесі бар делік.

Студент Айна AI201 пәніне тіркелгісі келеді. Жүйеде Айнаның студент екені, белгілі бір пәндерді меңгергені, бағалары және басқа академиялық мәліметтері сақталған.

Бірақ бізге бұл деректердің өзі жеткіліксіз.

Жүйе **«Айна AI201 пәніне тіркеле ала ма?»** деген сұраққа жауап беруі керек.

Ол үшін жүйе жай ғана деректерді сақтамай, олардың арасындағы байланысты түсінуі қажет.

Мысалы:

«Егер студент қажетті алдыңғы пәнді меңгерсе және AI201 пәнінде бос орын болса, сонда ғана ол пәнге тіркеле алады».

Міне, бұл жерде біз **деректен ережеге, ал ережеден қорытындыға** өтеміз.

Егер жүйе «Айна тіркеле алады» десе, біз одан:

«Неліктен? Қандай шарттар орындалды?»

деп сұрай аламыз.

Логикалық модельдің маңызды ерекшеліктерінің бірі — осындай қорытындының қалай алынғанын көрсету мүмкіндігі.

Сондықтан бүгінгі дәрістің негізгі сұрағы:

Жүйе белгілі фактілер мен ережелерге сүйене отырып, жаңа қорытындыны қалай шығарады?

Осы сұраққа дәріс барысында біртіндеп жауап беруге тырысамыз.

Бүгін біз логикалық модельді тек анықтама ретінде емес, толық жұмыс істейтін білімді ұсыну механизмі ретінде қарастырамыз.

Алдымен логикалық модельдің не екенін анықтайық.

Жалпы жолымыз мынадай болады:

табиғи тіл → формализация → фактілер мен ережелер → логикалық қорытынды → программалық іске асыру → нәтижені түсіндіру.

Алдымен бүгінгі тақырыптың өткен дәріспен байланысын анықтап алайық.

Өткен дәрісте білімді ұсынудың бірнеше моделін қарастырдық.

Семантикалық желі - объектілер арасындағы байланысты граф түрінде көрсетуге мүмкіндік береді.

Фреймдер объектінің құрылымын, қасиеттерін және атрибуттарын сипаттайды.

Онтологиялар белгілі бір пәндік саладағы ұғымдарды және олардың арасындағы қатынастарды жүйелейді.

Өнімдік ережелер білімді «ЕГЕР ... ОНДА ...» түрінде көрсетуге мүмкіндік береді.

Бұл тәсілдердің барлығы білімді құрылымдауға көмектеседі.

Ал логикалық модельдің ерекше қасиеті — білімді **формальды түрде сипаттап, одан логикалық салдар шығаруға мүмкіндік беруі**.

Сондықтан біз:

«Білімді қалай сақтаймыз?»

деген сұрақтан:

«Сақталған білімнен жаңа қорытындыны қалай аламыз?»

деген сұраққа өтеміз.

Логикалық модель — пәндік сала туралы білімді формальды тұжырымдар мен ережелер арқылы көрсетіп, сол білімнен логикалық қорытынды алуға мүмкіндік беретін білімді ұсыну моделі.

Бұл анықтамада үш маңызды ой бар.

Біріншісі — **білім**.

Жүйеде нақты пәндік сала туралы ақпарат болуы керек.

Екіншісі — **формальды ұсыну**.

Ақпарат еркін табиғи тілде емес, нақты құрылымы бар логикалық тілде беріледі.

Үшіншісі — **қорытындылау**.

Жүйе бастапқы біліммен шектелмей, одан жаңа білім шығара алады.

Мысалы, Студент Данияр туралы:

Passed(Daniyar, Programming1)

фактісі берілсін.

Ал ереже:

∀x (Passed(x, Programming1) → Eligible(x, Programming2))

түрінде жазылсын.

Мұнда **x** — кез келген студентті білдіретін айнымалы.

Егер Данияр Programming1 пәнін меңгерсе, жалпы ережені оған қолдану арқылы:

Eligible(Daniyar, Programming2)

деген жаңа қорытынды аламыз.

Бірақ бір маңызды нәрсені ұмытпау керек: логикалық қорытынды дұрыс болғанымен, бастапқы факт қате болса, нақты өмірдегі шешім де қате болуы мүмкін.

Сондықтан логика қорытындылаудың дұрыстығын қамтамасыз етеді, бірақ бастапқы деректің шынайылығын өзі тексермейді.

Осыдан келесі сұрақ туындайды: **дерек пен білімнің айырмасы қандай?**

Мысалы, жүйеде жай ғана:

85 саны тұр делік.

Контекст болмаса, оның нені білдіретінін білмейміз.

Ал:

Grade(Aruzhan, Programming1, 85)

деп жазсақ, ақпараттың мағынасы анықталады.

Бұл — Аружанның Programming1 пәнінен алған бағасы.

Егер университет ережесі бойынша 50 және одан жоғары балл пәннен өткенін білдірсе, онда осы деректен:

Passed(Aruzhan, Programming1)

деген жаңа білім алуға болады.

Сонымен:

Дерек — нақты объект немесе жағдай туралы берілген мәндер мен фактілер;

Білім — объектілердің қасиеттерін, қатынастарын, заңдылықтарын және қорытындылау ережелерін сипаттайтын құрылымдалған мазмұн.

Әдебиеттерде «дерек», «ақпарат» және «білім» ұғымдарының шекарасы әртүрлі түсіндірілуі мүмкін.

Сондықтан бұл айырмашылықты абсолютті деп емес, логикалық жүйені түсіндіруге көмектесетін жұмыс моделі ретінде қабылдаймыз.

Интеллектуалды жүйе үшін үш сұрақ маңызды:

Не белгілі?

Олар қалай байланысады?

Олардан қандай қорытынды шығады?

Осы сұрақтар бізді логикалық білім базасының үш негізгі элементіне алып келеді.

Логикалық білім базасында ақпараттың рөлі бірдей емес.

Оның бір бөлігі бастапқы жағдайды сипаттайды, екінші бөлігі байланыстарды анықтайды, ал үшінші бөлігі жүйеге қойылатын сұрақты көрсетеді.

Сондықтан үш ұғымды ажыратамыз.

Факт — білім базасында ақиқат деп қабылданған нақты тұжырым.

Мысалы:

Student(Aidana)

Бұл:

«Айдана— студент» дегенді білдіреді.

Ереже — белгілі шарттар орындалған кезде қандай қорытынды жасауға болатынын көрсетеді.

Мысалы:

Student(x) ∧ Passed(x, CS101) ∧ HasSeat(AI201) → Eligible(x, AI201)

Ал **сұраныс** — жүйеден нені білгіміз келетінін көрсетеді.

Мысалы:

Eligible(Aidana, AI201)?

Осы үш элементті қысқаша былай байланыстыруға болады:

ФАКТІЛЕР + ЕРЕЖЕЛЕР + ҚОРЫТЫНДЫЛАУ → СҰРАНЫСҚА ЖАУАП

Бірақ **Жүйе** факт пен ережені өңдеу үшін олардың қалай жазылатынын және нені білдіретінін білуі керек.

Сондықтан енді логикалық тілдің құрылымына көшеміз.

Логикалық модельдің жұмысын түсіну үшін осы үш ұғымды ажыратамыз:

Синтаксис — логикалық өрнектің дұрыс жазылу ережесі.

Мысалы:

Passed(Madina, CS101)

— белгілі құрылым бойынша жазылған өрнек.

Семантика — сол өрнектің мағынасы.

Біздің жағдайда:

«Мадина CS101 пәнін меңгерді».

Ал **инференция** — белгілі фактілер мен ережелер негізінде жаңа қорытынды алу процесі.

Егер:

Passed(Madina, CS101)

фактісі және:

Passed(x, CS101) → Eligible(x, AI201)

ережесі болса, онда:

Eligible(Madina, AI201)

деген қорытынды аламыз.

Есте сақтаудың қарапайым тәсілі: СҰРАҚ ҚОЮ АРҚЫЛЫ

Синтаксис — қалай жазамыз?
Семантика — нені білдіреді?
Инференция — одан не шығарамыз?

Енді логикалық өрнектердің ең қарапайым негізіне — пікірлерге көшеміз.

Пікір — ақиқат немесе жалған деп бағалауға болатын хабарлы сөйлем.

Мысалы:

«***Нұрлан бүгін дәріске қатысты***».

Бұл пікір, себебі оның ақиқат немесе жалған екенін тексеруге болады.

Ал:

«Дәріске кел!»

— бұйрық, сондықтан пікір емес.

Пікірлерді толық сөйлем түрінде қайта-қайта жазбау үшін оларды символдармен белгілейміз.

Мысалы:

P = «Нұрлан бүгін дәріске қатысты».

Классикалық пікірлер логикасында:

1 — ақиқат;

0 — жалған.

Бір ғана пікір қарапайым жағдайды сипаттайды. Ал нақты интеллектуалды жүйеде бірнеше шартты байланыстыру қажет.

Ол үшін логикалық амалдар қолданылады.

Бірінші амал — **терістеу**.

Ол:

¬P

түрінде жазылады және «P емес» дегенді білдіреді.

Екінші амал — **конъюнкция**:

P ∧ Q

Ол «P және Q» деп оқылады.

Конъюнкция екі шарт та орындалғанда ғана ақиқат.

Мысалы, Ернар зертханаға кіруі үшін:

P = «Ернар қауіпсіздік нұсқаулығынан өтті»;

Q = «Ернар қажетті қорғаныс құралын алды».

Онда кіру шарты:

P ∧ Q

болады.

Яғни AND қолданылған кезде **барлық қажетті шарт бірге орындалуы керек**.

Ал кейбір жағдайда барлық шарт емес, бірнеше мүмкіндіктің кемінде біреуі жеткілікті болады.

Бұл бізді дизъюнкцияға алып келеді.

Дизъюнкция:

P ∨ Q

түрінде жазылады және «P немесе Q» деп оқылады.

Классикалық логикадағы OR — қамтушы «немесе»: кемінде бір пікір ақиқат болса, нәтиже ақиқат.

Мысалы, Данияр университеттік жүйеге:

P — университеттік email арқылы

немесе

Q — студенттік ID арқылы

кіре алады делік.

Онда шарт:

P ∨ Q.

Екі тәсілдің кемінде біреуі жеткілікті.

Ал **эквиваленция**:

P ↔ Q

түрінде жазылып, «P сонда және тек сонда, егер Q» дегенді білдіреді.

Эквиваленция екі бағыттағы байланысты қамтиды:

P ↔ Q ≡ (P → Q) ∧ (Q → P)

Мысалы, Аружан жабық зертханаға **тек арнайы электрондық рұқсат алған жағдайда ғана** кіре алады және басқа кіру жолы қарастырылмаған делік. Сонда рұқсат пен кіру құқығы арасындағы байланысты екі бағытта сипаттауға болады.

Бірақ кез келген ережені эквиваленция деп қабылдауға болмайды.

Интеллектуалды жүйелердегі көптеген ережелер бір бағытта жұмыс істейді:

«Егер..., онда...»

Бұл — импликация.

Импликация:

P → Q

түрінде жазылады.

Мұнда:

P — алғышарт;

Q — қорытынды.

Мысалы:

P = «Әлихан барлық зертханалық жұмысты орындады».

Q = «Әлихан емтиханға жіберіледі».

Ереже:

P → Q.

Импликацияны міндетті түрде физикалық себеп-салдар ретінде түсінбеу керек. Ол ең алдымен белгілі шарт орындалғанда қандай қорытынды қабылданатынын көрсетеді.

Сонымен қатар импликация бір бағытты.

P → Q

берілсе, одан автоматты түрде:

Q → P

шықпайды.

Ал импликацияның ақиқаттық мәнін толық түсіну үшін шындық кестесін қарастырамыз.

Шындық кестесі бастапқы пікірлердің барлық мүмкін мәнінде логикалық өрнектің нәтижесін көрсетеді.

P	**Q**	**¬P**	**P ∧ Q**	**P ∨ Q**	**P → Q**	**P ↔ Q**
1	1	0	1	1	1	1
1	0	0	0	1	**0**	0
0	1	1	0	1	1	0
0	0	1	0	0	1	1

Мұнда импликацияға назар аударайық.

P → Q тек P ақиқат, Q жалған болғанда ғана жалған.

Мысалы, оқытушы Жансаяға:

«Егер жұмысты бүгін тапсырсаң, бүгін тексеремін»

деді делік.

P — Жансая жұмысты тапсырды.

Q — оқытушы жұмысты тексерді.

Егер Жансая тапсырса, бірақ оқытушы тексермесе:

1 → 0 = 0.

Шарт дәл осы жағдайда бұзылады.

Ал P орындалмаса, берілген импликацияның алғышарты іске қосылған жоқ.

Шындық кестесі бізге логикалық өрнектерді формальды тексеруге мүмкіндік береді.

Бірақ нақты жүйеге білім P және Q түрінде дайын келмейді. Ол табиғи тілде беріледі.

Сондықтан келесі кезең — **формализация**.

Формализация — табиғи тілдегі білімді нақты формальды құрылымға көшіру процесі.

Мысалы:

«Егер Мирас барлық міндетті тапсырмаларды орындаса және оның рейтингі 50 балдан төмен болмаса, ол емтиханға жіберіледі».

Белгілейік:

P — барлық тапсырма орындалды;

Q — рейтинг 50-ден төмен емес;
R — емтиханға жіберілді.
Сонда:
(P ∧ Q) → R
аламыз.
Формализация кезінде бастапқы ереженің мағынасын өзгертпеу маңызды.
Сондықтан үш сұрақ қоямыз:
Қандай объектілер бар?
Олардың арасында қандай қатынас бар?
Қандай шарт қандай қорытындыға әкеледі?
Бірақ мұнда пікірлер логикасының шектеуін көреміз.
Егер мың студент болса, әр студентке жеке P, Q және R белгілеу тиімсіз.
Бізге жалпылау қажет.
Сондықтан **предикаттар логикасына** өтеміз.
Бірінші ретті предикаттар логикасы объектілерді, олардың қасиеттерін және объектілер арасындағы қатынастарды көрсетуге мүмкіндік береді.
Мысалы:
Student(Sanzhar)
— «Санжар — студент».
Мұнда тұжырым тұтас P символымен жасырылмайды.
Student қасиетті көрсетеді, ал Sanzhar — нақты объект.
Бір құрылымды басқа объектілерге де қолдануға болады:
Student(Sanzhar)
Student(Aigerim)
Student(Alikhan)
Осылайша предикаттар логикасы білімді жалпылауға мүмкіндік береді.
Енді осы формулалардың элементтерін нақты ажыратып алайық.
Мысалы:
Studies(Asem, AI201)
формуласын қарастырайық.
Studies — **предикат**.
Предикат объектінің қасиетін немесе объектілер арасындағы қатынасты сипаттайды.
Asem және AI201 — **тұрақтылар**.
Тұрақты нақты объектіні білдіреді.
Ал:
Studies(x, AI201)
деп жазсақ, x — **айнымалы**.
Оның орнына әртүрлі студенттер келе алады.
Демек:
тұрақты — нақты объектіні;
айнымалы — мүмкін объектіні;
предикат — қасиетті немесе қатынасты көрсетеді.
Предикаттың қанша объектімен жұмыс істейтінін анықтау үшін тағы бір ұғым қажет.
Предикаттың арлығы — оның аргументтерінің саны.
Мысалы:
Teacher(Aliya)
— бір орынды предикат.
Ал:
Teaches(Aliya, Java)
— екі орынды предикат.

Предикаттар логикасында **функциялар** да кездеседі.

Функция белгілі аргументтер бойынша объектіні көрсетеді.

Мысалы:

AdvisorOf(Bekzat)

Бекзаттың академиялық кеңесшісін білдірсін.

Сонда:

Teacher(AdvisorOf(Bekzat))

— «Бекзаттың академиялық кеңесшісі — оқытушы».

Қысқаша:

Предикат тұжырым береді, функция объектіні көрсетеді.

Енді Student(x) формуласындағы x кімге қатысты екенін көрсету керек.

Ол үшін кванторлар қолданылады.

Негізгі екі квантор бар:

∀ — жалпылық кванторы;

∃ — бар болу кванторы.

Мысалы:

∀x (Student(x) → HasID(x))

дегеніміз:

«Әрбір студенттің ID-і бар».

Ал:

∃x (Student(x) ∧ Winner(x))

дегеніміз:

«Жеңімпаз болған кемінде бір студент бар».

Жалпы ережелерде ∀ кванторы импликациямен(→) жиі кездеседі, ал объектінің бар екенін көрсеткенде ∃ кванторы конъюнкциямен(∧) жиі қолданылады.

Бірақ бұл абсолютті ереже емес, жиі қолданылатын модельдеу үлгісі.

Кванторларды дұрыс таңдаумен қатар олардың орналасу реті де маңызды.

Салыстырайық:

∀x ∃y Helps(x, y)

және

∃y ∀x Helps(x, y)

Бірінші формула:

«Әрбір студентке көмектесетін кемінде бір адам бар».

Екінші формула:

«Барлық студентке көмектесетін бір адам бар».

Мысалы, Ернар, Алина және Дәуреннің әрқайсысына әртүрлі тьютор көмектессе, бірінші тұжырым орындалуы мүмкін.

Бірақ барлығына ортақ бір тьютор болмаса, екінші тұжырым орындалмайды.

Демек:

кванторлардың орнын өзгерту формуланың мағынасын өзгерте алады.

Сондықтан формуланы жазғаннан кейін оны табиғи тілге қайта аударып оқу пайдалы.

Кванторларға терістеу қолданғанда да мағынаға ерекше назар аудару керек.

Кванторларды терістеу

Мына тұжырымды алайық:

«Барлық студент тапсырманы орындады».

Оның жалпы формасы:

∀x P(x)

болсын.

Егер:

«Барлық студент тапсырманы орындады деген дұрыс емес»

десек, бұл кемінде бір студент тапсырманы орындамағанын білдіреді.
Сондықтан:
¬∀x P(x) ≡ ∃x ¬P(x)
Ал:
¬∃x P(x) ≡ ∀x ¬P(x)
Мысалы, Жанель тапсырманы өткізбесе, бір ғана қарсы мысал:
«Барлық студент тапсырманы өткізді»
деген жалпы тұжырымды терістеуге жеткілікті.
Енді білімді формальды түрде сипаттай аламыз.
Келесі міндет — сол білімнен **қорытынды алу**.

Modus Ponens

Негізгі қорытындылау ережелерінің бірі — **Modus Ponens**.
Оның формасы:
P → Q
P
∴ Q
Мысалы:
Submitted(Temirlan, Project) → Admitted(Temirlan, Defense)
ережесі бар.
Сонымен бірге:
Submitted(Temirlan, Project)
фактісі белгілі.
Онда:
Admitted(Temirlan, Defense)
деген қорытынды аламыз.
Яғни жаңа білім факт пен ереженің бірге қолданылуы нәтижесінде пайда болды.
Ал егер Q орындалмағаны белгілі болса, P туралы қорытынды жасауға болатын жағдай бар.
Бұл — Modus Tollens.

Modus Tollens

Modus Tollens құрылымы:
P → Q
¬Q
∴ ¬P
Мысалы:
«Егер Саматтың өтінімі дұрыс бекітілсе, растау коды пайда болады».
P — өтінім дұрыс бекітілді.
Q — растау коды пайда болды.
Егер растау кодының пайда болмағаны нақты белгілі болса, берілген ереже шеңберінде:
¬P
деген қорытынды жасаймыз.
Бірақ:
P → Q
және
Q
берілгенде:
P
деп қорытынды жасау жалпы жағдайда дұрыс емес.
Себебі Q басқа себеппен де орындалуы мүмкін.

Енді ережелерде нақты студенттің орнына x сияқты айнымалы тұрса, оны нақты фактпен сәйкестендіру қажет.

Ол үшін унификация қолданылады.

Унификация

Мысалы:

Passed(x, CS101) → Eligible(x, AI201)

ережесі бар.

Ал факт:

Passed(Kamila, CS101)

болсын.

Жүйе:

x = Kamila

алмастыруын тауып, ережені Камилаға қолданады.

Бұл процесс **унификация** деп аталады.

Унификация — екі логикалық өрнекті сәйкестендіретін айнымалылар алмастыруын табу процесі.

Оның арқасында әр студентке жеке ереже жазудың қажеті жоқ.

Енді жүйе фактілерден бастап қорытындыға қарай жүре алады немесе керісінше, мақсаттан бастап қажетті фактілерді іздей алады.

Осыдан екі негізгі стратегия шығады.

Forward Chaining

Forward Chaining — фактілерден бастап жаңа қорытындыларға қарай қозғалу.

Жалпы бағыты:

факт → ереже → жаңа факт → келесі ереже → қорытынды.

Мысалы:

Completed(Nursultan, Practice)

фактісі және:

Completed(x, Practice) → Ready(x, Project)

ережесі берілсін.

Осыдан:

Ready(Nursultan, Project)

шығады.

Егер бұл жаңа факт басқа ереженің шарты болса, қорытындылау ары қарай жалғасады.

Forward Chaining:

«Осы фактілерден тағы не шығады?»

деген сұраққа жауап береді.

Ал егер бізге барлық салдар емес, бір нақты сұрақтың жауабы ғана қажет болса, іздеуді мақсаттан бастаған тиімді.

Backward Chaining

Backward Chaining — мақсаттан бастап, оны дәлелдеуге қажетті фактілерге қарай қозғалу.

Мысалы:

Eligible(Amina, Internship)?

деген сұраныс бар.

Ереже:

PassedInterview(x) ∧ CompletedCourses(x) → Eligible(x, Internship)

болсын.

Жүйе:

PassedInterview(Amina)?

және

CompletedCourses(Amina)?

деген ішкі мақсаттарды тексереді.

Екеуі дәлелденсе, бастапқы мақсат та дәлелденеді.

Сондықтан айырмашылық:

Forward Chaining: «Осы фактілерден не шығады?»

Backward Chaining: «Осы мақсатты дәлелдеу үшін не керек?»

Мұндай ережелерді компьютерге тиімді өңдету үшін олардың құрылымын стандарттау қажет.

Осы жерде Horn clause ұғымына келеміз.

Horn clause

Horn clause — формальды түрде ең көбі бір оң литералы бар клауза.

Практикалық тұрғыдан біз оны көбіне:

$$A_1 \land A_2 \land ... \land A_n \rightarrow B$$

түріндегі ереже ретінде кездестіреміз.

Мысалы:

$$Student(Arman) \land PaidTuition(Arman) \rightarrow Active(Arman)$$

Яғни Арман студент болса және қажетті төлем шарты орындалса, оның мәртебесі белсенді деп анықталады.

Horn ережелері компьютерлік өңдеуге ыңғайлы болғандықтан, Prolog және Datalog сияқты логикалық программалау жүйелерінде кең қолданылады.

Ал логикалық дәлелдеудің жалпы әдістерінің бірі — **резолюция**.

Резолюция

Бұл дәрісте резолюцияның күрделі формальды түрлендірулерін қарастырмаймыз. Бізге негізгі идеясын түсіну жеткілікті.

Резолюция — логикалық формулалардан жаңа салдар алуға және белгілі тұжырымды қайшылық арқылы дәлелдеуге қолданылатын қорытындылау әдісі.

Мысалы, біз:

Eligible(Dana, Exam)

тұжырымын дәлелдегіміз келеді.

Оның терістеуін уақытша қабылдаймыз:

¬Eligible(Dana, Exam)

Егер білім базасындағы фактілер мен ережелерден:

Eligible(Dana, Exam)

міндетті түрде шығатын болса, онда:

Eligible(Dana, Exam)

және

¬Eligible(Dana, Exam)

бірге пайда болады.

Бұл — қайшылық.

Демек, мақсаттың терістеуі бастапқы біліммен үйлеспейді.

Енді осы теориялық ұғымдардың программалау тілінде қалай көрінетінін қарастырамыз.

Prolog

Prolog — фактілер, ережелер және сұраныстарға негізделген логикалық программалау тілі.

Мысалы:

```
student(aliya).
passed(aliya, cs101).
seat(ai201).

eligible(X, ai201) :-
   student(X),
   passed(X, cs101),
```

seat(ai201).

Алғашқы үш жол — фактілер.

Соңғы құрылым — ереже.

Енді:

?- eligible(aliya, ai201).

сұранысын орындасақ, қажетті фактілер табылған жағдайда:

true.

жауабын аламыз.

Бұл жерде біз бұған дейін қарастырған ұғымдардың барлығын бірге көріп тұрмыз:

факт → ереже → сұраныс → унификация → қорытынды.

Бірақ Prolog-тағы false жауабын дұрыс түсіндіру өте маңызды.

Prolog-та false нені білдіреді?

Мақсат туралы қажетті фактілер базаға енгізілмеген делік.

Сұраныс:

?- eligible(maksat, ai201).

Prolog:

false.

деп жауап беруі мүмкін.

Бұл жауапты:

«Мақсат нақты өмірде AI201 пәніне жарамсыз»

деп автоматты түрде түсіндіруге болмайды.

Операциялық тұрғыдан:

қазіргі программадағы фактілер мен ережелер арқылы сұраныстың дәлелі табылмады

деген мағына береді.

Мүмкін, Мақсат қажетті пәнді меңгерген, бірақ оның нәтижесі базаға әлі енгізілмеген шығар.

Сондықтан:

дәлелдің табылмауы мен классикалық жалғандық бір ұғым емес.

Осы айырмашылық толық емес біліммен жұмыс істегенде өте маңызды.

Ашық және жабық әлем жорамалдары

Ашық әлем жорамалы — Open World Assumption бойынша фактінің білім базасында болмауы оның жалған екенін білдірмейді.

Мысалы, Әділеттің ағылшын тілін білетіні туралы базаға ақпарат енгізілмеген делік.

Ашық әлем жағдайында:

«Әділет ағылшын тілін білмейді»

деп қорытынды жасамаймыз.

Жауап:

«Ақпарат жеткіліксіз».

Ал **Closed World Assumption — жабық әлем жорамалы** белгілі бір толық деп қабылданған база аясында дәлелденбеген тұжырымды жалған ретінде өңдеуге мүмкіндік береді.

Prolog-тағы:

\+ Goal

құрылымы **negation as failure**, яғни дәлелдеудің сәтсіздігі арқылы терістеу принципімен жұмыс істейді.

Сондықтан оның операциялық мінез-құлқын классикалық терістеумен толық теңестіруге болмайды.

Ақпараттың болмауы бір мәселе болса, бір-біріне қарама-қайшы ақпараттың қатар келуі — басқа мәселе.

Қайшылықты білім

Мысалы:

Passed(Rauan, CS101)

және:

¬Passed(Rauan, CS101)

бір білім базасында қатар тұр делік.

Бір тұжырым Рауан пәннен өткенін, екіншісі өтпегенін көрсетеді.

Бұл — **қайшылық**.

Классикалық логикада P және ¬P қатар қабылданған қайшылықты база өте күрделі мәселе тудырады: классикалық жарылыс принципі бойынша мұндай қайшылықтан кез келген тұжырым шығаруға болады.

Сондықтан нақты интеллектуалды жүйеде қайшылықтың себебін анықтау қажет.

Мүмкін, бір ақпарат ескі, екіншісі жаңартылған шығар.

Мүмкін, олардың дереккөздерінің сенімділігі әртүрлі шығар.

Осыдан маңызды қорытынды шығады:

Логикалық қорытындының сапасы бастапқы білімнің сапасына да тәуелді.

Логика біз берген біліммен жұмыс істейді. Ол қате бастапқы фактіні өздігінен түзетпейді.

Логикалық модельдердің артықшылықтары мен шектеулері

Логикалық модельдердің негізгі артықшылықтарының бірі — **анықтық**.

Факт пен ереже формальды түрде берілгендіктен, қорытындының қалай алынғанын тексеруге болады.

Екінші артықшылық — **түсіндірмелілік**.

Жүйе белгілі бір шешімнің қандай фактілер мен ережелерге сүйенгенін көрсете алады.

Үшінші артықшылық — **жалпылау**.

Айнымалылар мен кванторлар бір ережені көптеген объектіге қолдануға мүмкіндік береді.

Бірақ шектеулері де бар.

Нақты әлемдегі барлық білімді «ақиқат» және «жалған» деп қатаң бөлу оңай емес. Белгісіздік, ықтималдық, уақыт бойынша өзгеретін ақпарат және ережеден ерекше жағдайлар кездеседі.

Мысалы, Ботаның:

«академиялық белсенділігі жоғары»

деген тұжырымын логикалық жүйеге енгізу үшін алдымен «жоғары» деген ұғымның нақты критерийін анықтау керек.

Сондықтан қазіргі интеллектуалды жүйелерде логикалық әдістер басқа тәсілдермен, соның ішінде статистикалық модельдермен және машиналық оқытумен бірге қолданылуы мүмкін.

Логикалық модельдің негізгі күші өзгермейді:

ол білімді анық көрсетіп, қорытындының қандай негізде алынғанын бақылауға мүмкіндік береді.

Логикалық модельді құрудың жалпы жолы

Енді дәрістің барлық бөлігін бір процеске біріктірейік.

Логикалық модельді құру кезінде алдымен **пәндік саланы** анықтаймыз.

Одан кейін қажетті **объектілерді, қасиеттерді және қатынастарды** белгілейміз.

Келесі қадамда бастапқы **фактілерді** анықтаймыз.

Табиғи тілдегі ережелерді **формализациялаймыз**.

Қажетті **предикаттарды, тұрақтыларды, айнымалыларды және кванторларды** таңдаймыз.

Одан кейін қорытындылау механизмін қолданамыз.

Соңында алынған жауаптың:

қандай фактіге және қандай ережеге сүйенгенін тексереміз.

Мысалы, Зере ғылыми стажировкаға қатысқысы келеді делік.

Жүйеде оның студент екені, қажетті курсты аяқтағаны және стажировкада орын бар екені белгілі.

Ереже осы шарттардың барлығы орындалғанда студенттің стажировкаға жарамды екенін көрсетеді.

Жүйе:

«Зере жарамды»

деген қорытынды жасағанда біз тек жауапты емес, оның негізін де көре аламыз.

Осылайша бүкіл процесті:

дерек → формализация → білім → ереже → қорытынды → түсіндіру

түрінде көрсетуге болады.

Қорытынды

Сонымен, бүгін біз логикалық модельді жеке формулалардың жиынтығы ретінде емес, **білімді формальды түрде ұсыну және сол білімнен негізделген қорытынды алу механизмі** ретінде қарастырдық.

Пікірлер логикасы арқылы тұтас тұжырымдармен және:

¬, ∧, ∨, →, ↔

амалдарымен жұмыс істеуді үйрендік.

Одан кейін пікірлер логикасының мүмкіндігі жеткіліксіз болатын жағдайларды көріп, предикаттар логикасына өттік.

Предикаттар логикасында:

предикат, тұрақты, айнымалы, арлық, функция және кванторлар

арқылы білімді жалпылауды қарастырдық.

Содан кейін:

Modus Ponens, Modus Tollens, унификация, Forward Chaining, Backward Chaining және резолюция

арқылы білімнен жаңа қорытындының қалай алынатынын түсіндік.

Prolog мысалында теориялық білімнің программада қалай көрсетілетінін көрдік.

Соңында өте маңызды үш жағдайды ажыраттық:

ақпарат жоқ;

тұжырым жалған;

білімде қайшылық бар.

Бұл үш жағдай бір нәрсе емес.

Сондықтан логикалық модельді сауатты пайдалану — тек формуланы дұрыс жазу емес.

Біз:

бастапқы факт дұрыс па?

ереже дұрыс формализацияланған ба?

қорытынды қандай алғышарттан шықты?

ақпарат жеткілікті ме?

деген сұрақтарды да қоя білуіміз керек.

Дәрістің негізгі ойын бір сөйлеммен қорытындыласақ:

Логикалық модель жүйеге дайын жауапты бермейді; ол берілген білімнен жауапты қандай ережелер арқылы алуға болатынын формальды түрде көрсетеді.

Сондықтан интеллектуалды жүйенің жақсы жауабы:

«Не шешілді?»

деген сұрақпен аяқталмайды.

Ол:

«Неліктен дәл осындай шешім алынды?»

деген сұраққа да жауап беруі керек.

Сабақ соңындағы тапсырма

Енді дәрісті шағын тапсырмамен аяқтайық.

Айгерім университет кітапханасынан арнайы ғылыми басылым алғысы келеді.

Кітапхана ережесі:

«Егер пайдаланушы университет студенті болса және оның кітапханаға берешегі болмаса, онда ол ғылыми басылымды ала алады».

Осы жағдай бойынша:

1. Пәндік саладағы негізгі объектілерді анықтаңыздар.
2. Қажетті предикаттарды белгілеңіздер.
3. Табиғи тілдегі ережені логикалық формаға ауыстырыңыздар.
4. Айгерім туралы қажетті фактілерді жазыңыздар.
5. Жүйеге қойылатын сұранысты құрыңыздар.
6. Қандай қорытынды шығатынын көрсетіңіздер.
7. Бір қажетті факт базаға енгізілмесе, нәтижені қалай түсіндіретініңізді айтыңыздар.

Осы тапсырма бүгінгі дәрістің бүкіл логикасын бір жерде біріктіреді:

табиғи тіл → формализация → факт → ереже → сұраныс → қорытынды → түсіндіру.

Егер осы тізбектің әрбір кезеңін өз сөздеріңізбен түсіндіре алсаңыздар, онда логикалық модельдердің негізгі идеясын меңгердіңіздер.

Бүгін біз компьютерге білімді қалай формальды түрде беруге және сол білімнен негізделген қорытынды алуға болатынын қарастырдық.

Келесі тақырыптарда осы білім бізге интеллектуалды жүйелердің күрделірек шешім қабылдау механизмдерін түсінуге негіз болады.