7-дәріс.
Логикалық шығару механизмдері
Дәрістің мақсаты
Бүгін біз интеллектуалды ақпараттық жүйелердегі **логикалық шығару механизмдерінің** жұмыс істеу принциптерін тереңірек қарастырамыз.
Алдыңғы дәрісте **сараптамалық жүйенің архитектурасымен** танысып, **білім базасы**, **жұмыс жадысы** және **инференция механизмінің** өзара байланысын түсіндік.
Енді мынадай сұраққа жауап іздейміз:
компьютер белгілі фактілер мен ережелердің негізінде жаңа білімді қалай дәлелдейді және дұрыс қорытындыны қалай таңдайды?
Дәріс барысында логикалық қорытындының математикалық негіздерін, **Modus Ponens** және **Modus Tollens** ережелерін, **тура және кері шығару алгоритмдерін**, унификацияны, резолюцияны, ережелер арасындағы қайшылықтарды шешуді және белгісіз ақпаратпен жұмыс істеу ерекшеліктерін зерттейміз.
1. Логикалық шығару механизмдерінің пайда болуы және маңызы
Алдымен логикалық **шығару механизмдері** не үшін қажет болғанын қарастырайық.
Адам күнделікті өмірде белгілі ақпаратты пайдаланып, жаңа қорытынды жасайды.
Мысалы
біз белгілі бір пәнге тіркелу үшін алдыңғы пәнді меңгеру қажет екенін білеміз.
Егер студент сол пәнді сәтті аяқтаса және тіркелуге басқа кедергі болмаса, оның келесі пәнге тіркеле алатынын анықтаймыз.
Адам үшін мұндай ойлау табиғи көрінеді. Бірақ компьютерге қорытынды шығару үшін нақты ережелер мен алгоритмдер қажет.

Логикалық қорытынды жасаудың ғылыми негіздері ежелгі грек философиясынан басталады. Аристотель силлогизмдерді зерттеп, белгілі алғышарттардан жаңа тұжырым алудың формаларын сипаттаған.
Силлогизм — бұл екі бастапқы пайымдаудан (алғышарттан) жаңа пайымдау (қорытынды) шығарылатын логикалық ойлау нысаны.
мысал (логикалық тұрғыдан дұрыс):
Барлық металлдар **электр тоғын өткізеді.** Мыс — **металл**. Демек, мыс электр **тоғын өткізеді**.
Силлогизмнің құрылымы
Кез келген силлогизмде үш термин болады:
Үлкен термин — қорытындының предикаты («өткізеді»).
Кіші термин — қорытындының субъектісі («мыс»).
Ортаңғы термин — екі алғышартта да бар, бірақ қорытындыда жоқ байланыстырушы буын («металл»).

Силлогизмнің негізгі түрлері

Түрі	Мысалы
Категориялық	Барлық А ол В; С ол А; демек, С ол В.
Шартты (гипотетикалық)	Егер жаңбыр жауса, жер дымқыл болады. Жаңбыр жауып тұр. Демек, жер дымқыл.
Бөлшектік (дизъюнктивтік)	А ол не В, не С. А ол В емес. Демек, А ол С.

XIX ғасырда Джордж Буль логикалық операцияларды алгебралық түрде көрсетуге мүмкіндік беретін математикалық логиканы дамытты.
Мысал: Буль алгебрасында $1 + 0 = 1$ (шын НЕМЕСЕ жалған = шын) $1 \times 0 = 0$ (шын ЖӘНЕ жалған = жалған) Мұнда 1 — «шын», 0 — «жалған» мәнін білдіреді.

Кейін Готлоб Фреге предикаттар логикасының негіздерін қалыптастырды.
Мысал: «Барлық х үшін: егер х — мысық болса, онда х — жануар.» Яғни: ∀x (Мысық(x) → Жануар(x)).
XX ғасырда математикалық логика есептеу техникасымен тығыз байланыса бастады.
Мысал: Компьютер бағдарламасы шартты тексереді: ЕГЕР баға >= 50 ОНДА "өтті" ӘЙТПЕСЕ "құлады" Бұл — логикалық операцияны есептеуге қолдану.
Автоматтандырылған теорема дәлелдеу, логикалық программалау және жасанды интеллект салаларында қорытынды шығару алгоритмдері кеңінен зерттелді.
Автоматтандырылған теорема дәлелдеу мысалы: Бағдарламаға береміз: «Барлық шаршы — тіктөртбұрыш. Барлық тіктөртбұрыштың төрт бұрышы бар.» Бағдарлама өзі қорытынды шығарады: «Шаршының төрт бұрышы бар.»
Логикалық программалау мысалы (Prolog):
prolog
ана(Айгүл, Нұрлан).
әке(Нұрлан, Ержан).
әже(X, Z) :- ана(X, Y), әке(Y, Z).
Запрос: ?- әже(Айгүл, Ержан). → **иә (true)**
Логика:

- X = Айгүл, Z = Ержан
- іздейміз Y: Айгүл — ана Y? → Y = Нұрлан ✓
- Нұрлан — әке Ержана? → иә ✓
- Онда, Айгүл — әже Ержана → **true**

Жасанды интеллект мысалы: Ауа райы AI жүйесі: «Егер аспан бұлтты ЖӘНЕ ылғалдылық 80%-дан жоғары болса, онда жаңбыр жаууы мүмкін.» Аспан бұлтты және ылғалдылық 85% → Жүйе қорытынды шығарады: «Жаңбыр ықтималдығы жоғары.»
1965 жылы Джон Алан Робинсон ұсынған резолюция принципі автоматтандырылған логикалық дәлелдеудің маңызды әдістерінің біріне айналды.
Резолюция принципі мысалы: Екі факті бар:

1. **«Барлық гүл суды қажет етеді.» → (Гүл(x) → Су(x))**
2. **«Раушан — гүл.» → Гүл(раушан)**

Резолюция әдісі бірінші тұжырымды теріске шығарып, екіншісімен біріктіреді: Қорытынды: **Су(раушан)** — «Раушан суды қажет етеді.»
Осы тарихи дамудың нәтижесінде компьютерге тек ақпарат сақтауды емес, формалды ережелер бойынша жаңа білім алуды үйретуге мүмкіндік туды.
Қорытынды мысал: Медициналық AI жүйесі:
Білім базасында:
«Егер қызу 38°C-тан жоғары болса, онда инфекция болуы мүмкін.» Науқастың қызуы 39°C → Жүйе жаңа білім шығарады: «Инфекция ықтималдығы бар.» Бұл — компьютердің тек сақтауды емес, **жаңа қорытынды жасауды** көрсетеді.

Негізгі анықтама

Логикалық шығару механизмі — белгілі фактілер мен логикалық ережелерге сүйене отырып, жаңа тұжырымдарды алу немесе берілген тұжырымның дұрыстығын тексеру үшін қолданылатын алгоритмдер мен әдістер жиынтығы.
Қарапайым түрде:
Бастапқы фактілер мен білім
Логикалық шығару алгоритмі
Жаңа қорытынды немесе дәлелдеу нәтижесі
Бұл механизм сараптамалық жүйелерде ғана емес, логикалық программалауда, білім графтарында, автоматтандырылған жоспарлауда, ережелерге негізделген басқару жүйелерінде және басқа да интеллектуалды технологияларда қолданылады.

2. Логикалық шығару мен қарапайым есептеудің айырмашылығы

Енді есептеу мен логикалық қорытындының айырмашылығын түсінейік.

Мысалы, компьютерге екі сан берсек:

a=15, b=25

Ол:

a+b=40

деп есептейді.

Бұл — арифметикалық операция.

Ал интеллектуалды жүйеде келесі ақпарат бар делік:

- Студент CS1 пәнін аяқтады.
- CS1 пәні CS2 пәнінің пререквизиті.
- CS2 пәніне тіркелу үшін пререквизитті орындау және бос орынның болуы қажет.
- CS2 пәнінде бос орын бар.

Компьютер осы мәліметтерді пайдаланып:

«Студент CS2 пәніне тіркелу талаптарын орындайды»

деген жаңа қорытынды шығара алады.

Мұнда компьютер жай ғана сандарды есептеп отырған жоқ. Ол фактілер арасындағы логикалық байланыстарды пайдаланып отыр.

Салыстыру	**Қарапайым есептеу**	**Логикалық шығару**
Кіріс ақпараты	Сандар, айнымалылар	Фактілер, ережелер
Негізгі әрекет	Арифметикалық операция	Логикалық ережені қолдану
Нәтиже	Есептелген мән	Жаңа тұжырым немесе дәлел
Мысал	15 + 25 = 40	Шарттар орындалса, тіркелуге рұқсат

Осыдан біз логикалық шығару механизмінің негізгі ерекшелігін көреміз: ол бұрыннан белгілі ақпаратты жаңа білім алу үшін пайдаланады.

3. Логикалық қорытындының негізгі элементтері

Логикалық шығару механизмін түсіну үшін төрт негізгі ұғымды ажыратып алайық.

Факт — жүйеде ақиқат деп қабылданған немесе расталған мәлімет.

Ереже — белгілі шарттар орындалған жағдайда қандай қорытынды жасауға болатынын сипаттайтын логикалық байланыс.

Алғышарт — қорытынды шығаруға негіз болатын тұжырым.

Қорытынды — алғышарттарға логикалық ережелерді қолдану нәтижесінде алынған жаңа тұжырым.

Мысалы:

Факт:

Completed(Аян, CS1)

Ереже:

ЕГЕР студент CS1 пәнін аяқтаса **ЖӘНЕ** CS1 пәні CS2 пәнінің пререквизиті болса,

ОНДА студент CS2 пәнінің пререквизит талабын орындайды.

Нәтиже:

PrerequisiteSatisfied(Аян, CS2) Алғышарт орындалды

Бірақ мұнда маңызды мәселе бар:

пререквизитті орындау пәнге тіркелуге толық рұқсат берілгенін білдірмейді.

Бос орын, академиялық шектеулер және басқа талаптар да болуы мүмкін.

Сондықтан білім базасындағы ережелерді дұрыс құру өте маңызды.

4. Логикалық шығару механизмдерінің формалды негізі

Енді логикалық қорытындыны математикалық тұрғыдан қарастырайық.

Математикалық логикада тұжырымдар әртүрлі символдармен белгіленеді.

Мысалы:

- P— студент пререквизитті орындады;
- Q— пәнде бос орын бар;
- R — студент тіркелу талаптарын орындайды.

Ережені былай жазуға болады:

$$(P \land Q) \to R$$

Мұндағы:

$\land$ — логикалық ЖӘНЕ;

$\to$ — импликация, яғни «егер ... болса, онда ...»;

P, Q, R — логикалық тұжырымдар.

Бұл өрнек:

Егер студент пререквизитті орындаса және пәнде бос орын болса, онда ол осы модельде қарастырылған тіркелу талаптарын орындайды дегенді білдіреді.

Ал P және Q ақиқат болса, R қорытындысын алуға болады.

Бірақ егер P белгісіз болса, біз оны автоматты түрде жалған деп қабылдай алмаймыз. Бұл білім базасының қандай логикалық болжамдармен жұмыс істейтініне байланысты.

Логикалық операциялар

Операция	Белгісі	Мағынасы
Конъюнкция	$P \land Q$	P және Q
Дизъюнкция	$P \lor Q$	P немесе Q
Терістеу	$\neg P$	P емес
Импликация	$P \to Q$	Егер P болса, онда Q
Эквиваленция	$P \leftrightarrow Q$	P және Q бірдей ақиқаттық мәнге ие

Бұл операциялар логикалық ережелердің негізін құрайды.

5. Modus Ponens — тікелей логикалық қорытынды

Логикалық шығару ережелерінің ең танымал түрлерінің бірі — Modus Ponens.

Оның құрылымы:

$$P \to Q$$

$$P$$

$$\therefore Q$$

Яғни:

Егер P орындалса, Q орындалады.

P орындалды.

Демек, Q орындалады.

Мысал

Университетте зертханалық жұмысқа қатысу үшін қауіпсіздік нұсқаулығынан өту міндетті делік.

Ереже:

ЕГЕР студент қауіпсіздік нұсқаулығынан өтсе, ОНДА оның қауіпсіздік бойынша бастапқы рұқсаты бар.

Факт:

Айбек қауіпсіздік нұсқаулығынан өтті.

Қорытынды:

Айбектің қауіпсіздік бойынша бастапқы рұқсаты бар.

Мұнда қорытындының дұрыстығы бастапқы ереженің және фактінің дұрыстығына байланысты.

Бірақ осыдан Айбектің зертханаға кез келген уақытта кіре алатынын қорытындылауға болмайды. Өйткені зертханаға кіру үшін басқа талаптар да болуы мүмкін.

Осы мысал бізге логикалық қорытындының дәл және шектеулі болуы керектігін көрсетеді.

6. Modus Tollens — терістеу арқылы қорытынды шығару

Екінші маңызды ереже — Modus Tollens.

Оның құрылымы:

$$P \to Q$$

$$\neg Q$$

$$\therefore \neg P$$

Яғни:
Егер P болса, Q болуы тиіс.
Бірақ Q орындалмайды.
Демек, P орындалмайды.

Мысал

Бізде келесі ереже бар делік:
Егер сервердің негізгі қызметі толық жұмыс істеп тұрса, оның міндетті денсаулық тексерісі сәтті аяқталады.
Жүйеде денсаулық тексерісі сәтсіз аяқталғаны анықталды.
Осы ереже дұрыс және оның шарттары толық сақталған жағдайда:
Сервердің негізгі қызметі толық жұмыс істеп тұр деген тұжырымды қабылдай алмаймыз.
Нақты логикалық формада:

$$HealthyServer \to HealthCheckPassed$$

$$\neg HealthCheckPassed$$

$$\therefore \neg HealthyServer$$

Бұл әдіс диагностикада пайдалы.
Бірақ бір нәрсеге назар аударайық: тексеріс нәтижесінің сенімділігі және ереженің дұрыстығы өте маңызды. Егер тексеріс құралының өзі істен шықса, нақты өмірдегі қорытындыны қосымша тексеру қажет.

Екі ереженің айырмашылығы

Ереже	Бастапқы ақпарат	Қорытынды
Modus Ponens	$P \to Q, P$	Q
Modus Tollens	$P \to Q, \neg Q$	$\neg P$

7. Дұрыс және қате логикалық қорытынды

Логикалық шығару кезінде жиі кездесетін қателерді қарастырайық.
Мысалы:
Егер студент олимпиада жеңімпазы болса, оған арнайы сертификат беріледі.
Бұл ережені былай белгілейміз:

$$P \to Q$$

Мұнда:
P— студент олимпиада жеңімпазы;
Q — студентте арнайы сертификат бар.
Енді студентте сертификат бар екені белгілі болсын.
Осыдан оның міндетті түрде олимпиада жеңімпазы екенін айтуға бола ма?
Жоқ.
Өйткені сертификат басқа негізбен де берілуі мүмкін.
Бұл қате салдарды растау қатесі деп аталады.

$$P \to Q, \quad Q$$

осыдан P міндетті түрде шықпайды.
Тағы бір қате:
Студент олимпиада жеңімпазы емес, демек сертификат алмайды.
Бұл да дұрыс емес. Өйткені студент басқа жетістігі үшін сертификат алуы мүмкін.
Мұндай қателер интеллектуалды жүйелерде өте қауіпті, себебі қате қорытынды кейінгі ережелерге жаңа факт ретінде беріліп, бірнеше қате шешімге әкелуі мүмкін.

8. Тура логикалық шығару алгоритмі

Алдыңғы дәрісте Forward Chaining ұғымымен таныстық. Енді оның ішкі алгоритмін тереңірек зерттейік.
Тура логикалық шығару — бастапқы фактілерден басталып, қолданылатын ережелерді біртіндеп іске қосу арқылы жаңа қорытындылар алу әдісі.
Негізгі идеясы:
Бізде қандай фактілер бар және осы фактілердің негізінде тағы не анықтай аламыз?

Алгоритм

1. Бастапқы фактілер жұмыс жадысына енгізіледі.
2. Білім базасындағы барлық ережелердің шарттары тексеріледі.
3. Шарттары орындалған ережелер анықталады.
4. Қажет болса, олардың арасынан орындалатын ереже таңдалады.
5. Ереженің қорытындысы жұмыс жадысына қосылады.
6. Жаңа фактілерге байланысты ережелер қайта тексеріледі.
7. Жаңа қорытынды алынбайтын немесе тоқтату шарты орындалатын сәтке дейін процесс жалғасады.

Күрделірек мысал

Интеллектуалды жүйе сервердің жұмыс жағдайын бақылайды.
Бастапқы фактілер:

Факт	**Мән**
CPU жүктемесі	94%
Жауап беру уақыты	850 мс
Белсенді пайдаланушылар	1200
Қолжетімді сервер саны	3

Ережелер:
R1:
ЕГЕР CPU > 90% **ЖӘНЕ** жауап беру уақыты > 700 мс **БОЛСА**,
ОНДА ServerLoad = Critical.
R2:
ЕГЕР ServerLoad = Critical **ЖӘНЕ** белсенді пайдаланушылар > 1000 **БОЛСА**,
ОНДА ScalingNeeded = Yes. Кеңейту қажет
R3:
ЕГЕР ScalingNeeded = Yes **ЖӘНЕ** қолжетімді сервер саны ≥ 2 **БОЛСА**,
ОНДА RecommendLoadBalancing = Yes. Жүктемені теңестіруді ұсыну

Енді ережелердің қалай орындалатынын көрейік.
БАСТАПҚЫ ФАКТІЛЕР
CPU = 94%, ResponseTime = 850 мс, Users = 1200, Servers = 3
R1 іске қосылады
ServerLoad = Critical Сервер жүктемесі = Критикалық

R2 іске қосылады
ScalingNeeded = Yes Кеңейту қажет

R3 іске қосылады
RecommendLoadBalancing = Yes Жүктемені теңестіруді ұсыну

Нәтижесінде жүйе жүктемені теңестіруді ұсынады.
Бұл жерде **R3** ережесі бастапқы фактілермен тікелей іске қосылған жоқ. Оған **R1** және **R2** ережелері арқылы алынған аралық қорытындылар қажет болды.
Осы ерекшелік тура логикалық шығарудың негізгі артықшылығын көрсетеді: бірнеше ережені тізбектеп қолдану арқылы күрделі қорытынды алуға болады.

9. Кері логикалық шығару алгоритмі

Енді осы есепті басқа жағынан қарастырайық.
Бізге барлық ықтимал қорытындылар қажет емес делік.
Тек:
«**Жүктемені теңестіру қажет пе?»** деген сұраққа жауап керек.
Мұндай жағдайда кері логикалық шығару тиімді болуы мүмкін.
Кері логикалық шығару — берілген мақсаттан басталып, оны дәлелдеу үшін қажетті ережелер мен фактілерді іздеу әдісі.
Жүйе алдымен:
RecommendLoadBalancing = Yes (Жүктемені теңестіруді ұсыну**)**
мақсатын алады.
Содан кейін осы қорытындыны беретін **R3** ережесін табады.
R3 үшін:
ScalingNeeded = Yes (Масштабтау қажет**)**
және сервер саны кемінде екі болуы қажет.
Сервер саны туралы факт бар. Бірақ **ScalingNeeded** мәнін дәлелдеу керек.
Оны R2 ережесі береді.
R2 үшін ServerLoad = Critical (**Сервер жүктемесі = Критикалық**)болуы керек.
Ал оны R1 ережесі арқылы дәлелдеуге болады.
Сөйтіп жүйе мақсаттан бастап, бастапқы фактілерге дейін кері бағытта жүреді.
Мақсат: RecommendLoadBalancing = Yes **(**Жүктемені теңестіруді ұсыну**)**
R3 шарттары: ScalingNeeded = Yes және Servers ≥ 2
R2 шарттары: ServerLoad = Critical және Users > 1000
R1 шарттары: CPU > 90% және ResponseTime > 700 мс
Фактілер тексеріледі → мақсат расталады
Кері шығаруда жүйе тек мақсатқа қажетті фактілерді іздеуге тырысады.
Сондықтан нақты гипотезаны тексеру кезінде артық ережелерді қарастыруды азайтуы мүмкін.

10. Тура және кері шығару механизмдерінің тиімділігі

Екі тәсіл де логикалық тұрғыдан маңызды. Бірақ олардың қолданылу жағдайлары әртүрлі.

Критерий	**Тура шығару**	**Кері шығару**
Басталу нүктесі	Фактілер	Мақсат
Негізгі бағыт	Деректерден қорытындыға	Қорытындыдан қажетті шарттарға
Нәтиже	Бірнеше ықтимал қорытынды	Нақты мақсаттың дәлелденуі
Тиімді қолданылуы	Мониторинг, оқиғаларды өңдеу	Диагностика, сұранысқа жауап
Әлсіз жағы	Қажетсіз қорытындылар шығаруы мүмкін	Мақсат көп болса, тексеру күрделенуі мүмкін

Кейбір интеллектуалды жүйелерде екі әдіс біріктіріледі.
Мысалы, жүйе алдымен тура шығару арқылы сервердің жалпы жағдайын бағалайды. Ал нақты ақаудың себебін анықтау қажет болғанда кері шығаруды қолданады.
Мұндай тәсіл аралас логикалық шығару деп аталады.

11. Предикаттар логикасы және айнымалылар

Осы уақытқа дейін біз нақты объектілерге қатысты ережелерді қарастырдық. Бірақ интеллектуалды жүйеде мыңдаған студент, жүздеген пән немесе көптеген құрылғы болуы мүмкін.
Әр объект үшін бөлек ереже жазу тиімсіз.
Мысалы:

- Аян CS1 пәнін аяқтады.
- Айдана CS1 пәнін аяқтады.
- Дәурен CS1 пәнін аяқтады.

Әр студентке жеке ереже жазудың орнына жалпы ереже құруға болады.
Мұнда **предикаттар логикасы** қолданылады.
Предикат — объектілердің қасиеттерін немесе олардың арасындағы қатынастарды сипаттайтын логикалық өрнек.
Мысалы:

$$Completed(x, CS101)$$

Мұндағы x — студентті білдіретін айнымалы.
Егер барлық студентке ортақ ереже жазғымыз келсе:

$$\forall x \ (Completed(x, CS101) \rightarrow BasicReady(x))$$

Бұл өрнек:
Кез келген x студенті CS101 пәнін аяқтаса, ол осы модельде анықталған базалық дайындық талабын орындайды.
дегенді білдіреді.

Мұндағы $\forall$ белгісі «барлық» деген мағынаны білдіреді.

Ал $\exists$ белгісі «кемінде біреуі бар» дегенді білдіреді.

Мысалы:

$$\exists x \ Completed(x, CS101)$$

Бұл — CS101 пәнін аяқтаған кемінде бір студент бар деген тұжырым.
Предикаттар логикасы интеллектуалды жүйелерге нақты объектілермен ғана емес, объектілердің жалпы топтарымен жұмыс істеуге мүмкіндік береді.
12. Унификация механизмі
Енді предикаттар логикасында өте маңызды ұғымға келеміз.
Унификация — екі логикалық өрнекті айнымалылардың орнына сәйкес мәндерді қою арқылы бірдей ету процесі.

Мысалы, бізде:

$$Completed(x, CS101)$$

деген өрнек бар.

Ал жұмыс жадысында:

$$Completed(\text{Аян}, CS101)$$

фактісі сақталған.

Осы екі өрнекті сәйкестендіру үшін:

$$x = \text{Аян}$$

деп аламыз.

Сонда жалпы ереже нақты студентке қолданылады.

Күрделірек мысал

Білім базасында:

$$Prerequisite(x, y)$$

деген предикат бар.

Сонда жалпы ереже нақты студентке қолданылады.

Унификация логикалық программалауда және автоматтандырылған дәлелдеуде өте маңызды қызмет атқарады.
Ол компьютерге:
«Бұл жалпы ережені нақты қандай объектіге қолдануға болады?»
деген сұраққа жауап беруге көмектеседі.

13. Резолюция әдісі

Енді логикалық қорытынды жасаудың тағы бір маңызды әдісін қарастырайық.
Резолюция — логикалық тұжырымдарды клаузалар түрінде пайдаланып, бір-біріне қарама-қарсы литералдарды жою арқылы жаңа клаузалар алуға негізделген қорытынды шығару ережесі. **Клауза** — литералдардың **дизъюнкциясы** (НЕМЕСЕ арқылы байланысуы).
Бұл әдіс **автоматтандырылған теорема** дәлелдеуде кеңінен қолданылады.
Мысалы, бізде:

$$P \vee Q$$

және

$$\neg P \vee R$$

деген екі өрнек бар.
Бірінші өрнекте P, екіншісінде кездеседі. $\neg P$
Резолюция арқылы:

$$Q \vee R$$

деген жаңа өрнек алуға болады.

Резолюция арқылы дәлелдеу идеясы

Көптеген автоматтандырылған дәлелдеу жүйелерінде мақсатты тікелей дәлелдеудің орнына оның терістеуі қабылданады.
Егер білім базасы мен мақсаттың терістеуінен логикалық қайшылық шықса, бастапқы мақсат дәлелденген болып есептеледі.
Мысалы:

1. Барлық тіркелген пайдаланушылардың жүйеге кіру құқығы бар.
2. Аян тіркелген пайдаланушы.
3. Аянның жүйеге кіру құқығы бар екенін дәлелдеу қажет.

Біз мақсаттың терістеуін қабылдаймыз:
Аянның жүйеге кіру құқығы жоқ.
Бірақ алғашқы екі фактіден оның кіру құқығы бар екені шығады.
Демек, **қарама-қайшылық** пайда болады.
Сондықтан мақсат дәлелденеді.
Бұл тәсіл **қайшылыққа келтіру** арқылы дәлелдеу принципімен байланысты.
Резолюция әсіресе **бірінші ретті логикадағы** формалды дәлелдеу жүйелерінде маңызды.

14. Хорн клаузалары және логикалық программалау

Көптеген ережелерге негізделген жүйелерде Хорн клаузалары қолданылады.
Хорн клауза — ең көбі бір оң литералы бар клауза.
Практикада жиі қолданылатын түрі:

$$A \wedge B \rightarrow C$$

Бұл бізге таныс:
ЕГЕР А ЖӘНЕ В БОЛСА, ОНДА С
ережесіне сәйкес келеді.
Мысалы:

$$Completed(x, CS101) \wedge Available(CS201) \rightarrow Eligible(x, CS201)$$

Мұндай ережелер логикалық программалау тілдерінде ыңғайлы түрде жазылады.

Prolog тіліндегі мысал

```
completed(ayan, cs101).
available(cs201).

eligible(X, cs201) :-
   completed(X, cs101),
   available(cs201).
```

Бұл программада:

completed(ayan, cs101).

— факт.

available(cs201).

— факт.

Ал:

```
eligible(X, cs201) :-
   completed(X, cs101),
   available(cs201).
```

— ереже.

Егер біз:

?- eligible(ayan, cs201).

деп сұрасақ, жүйе:

true.

деп жауап береді.

Мұнда Prolog ереженің шарттарын тексеріп, мақсатты дәлелдейді.

Назар аударыңыздар: бұл мысалда тек көрсетілген екі талап ескерілген.

Нақты университеттің тіркеу жүйесінде басқа талаптар да болуы мүмкін.

15. Ережелер арасындағы қайшылықтар

Енді **логикалық шығару механизмдерінің** практикалық мәселелерінің бірін қарастырайық.

Кейде бір уақытта бірнеше ереже іске қосылуы мүмкін.

Мысалы, университеттің интеллектуалды жүйесінде:

R1: ЕГЕР студенттің GPA көрсеткіші 2.0-ден төмен болса, ОНДА академиялық консультация қажет.

R2: ЕГЕР студент академиялық демалыста болса, ОНДА әдеттегі академиялық консультацияны автоматты түрде тағайындамау керек.

Бір студенттің GPA көрсеткіші 1.8 және ол академиялық демалыста болуы мүмкін.

Бұл жағдайда екі ереже де іске қосылады.

Егер жүйе олардың арасындағы байланысты ескермесе, қарама-қайшы ұсыныстар беруі мүмкін.

Сондықтан қайшылықтарды **шешу стратегиялары** қажет.

Негізгі стратегиялар

Стратегия	**Сипаттамасы**
Басымдық	Маңызды ережеге жоғары приоритет беріледі
Нақтылық	Шарттары неғұрлым нақты ереже таңдалады
Жаңалық	Жаңа фактілермен байланысты ережеге басымдық беріледі
Белгіленген реттілік	Ережелер алдын ала бекітілген тәртіппен орындалады

Мысал

R1 басымдығы = 5.

R2 басымдығы = 10.

Егер жүйеде жоғары сан жоғары басымдықты білдірсе, **R2** бірінші қарастырылады.

Бірақ бұл **R1** ережесі міндетті түрде жойылады деген сөз емес. Нақты нәтиже ережелер қозғалтқышының жұмыс принципіне және жүйеде анықталған саясатқа байланысты.

16. Логикалық шығару циклдері және шексіз қайталану
Кейде ережелер бір-бірін қайта-қайта іске қосуы мүмкін.
Мысалы:
R1: ЕГЕР А БОЛСА, ОНДА В.
R2: ЕГЕР В БОЛСА, ОНДА А.
Егер жүйе жаңа және бұрыннан белгілі фактілерді ажыратпаса, бұл ережелер шексіз қайталануы мүмкін.
Сондықтан логикалық шығару механизмдерінде:

- бұрын алынған фактілерді бақылау;
- қайталанатын қорытындыларды анықтау;
- ереженің қайта іске қосылуын шектеу;
- тоқтату шарттарын анықтау

сияқты тәсілдер қолданылады.
Мысалы,
жүйе **А** және **В** фактілерін бір рет алғаннан кейін оларды қайтадан **жаңа факт** ретінде қоспауы мүмкін.
Бұл логикалық шығару процесінің аяқталуын қамтамасыз етуге көмектеседі.
17. Белгісіз және толық емес ақпарат
Нақты өмірде барлық ақпарат белгілі бола бермейді.
Мысалы,
студенттің GPA көрсеткіші бар, бірақ сабаққа қатысу мәліметі әлі енгізілмеген делік.
Біз:
«Сабаққа қатысу көрсеткіші белгісіз»
деген жағдайды:
«Сабаққа қатысуы төмен»
немесе
«Сабаққа қатысуы қалыпты»
деген қорытындылармен шатастырмауымыз керек.
Мұнда екі маңызды ұғым бар.
Ашық әлем болжамы (Open World Assumption) — білім базасында фактінің болмауы оның жалған екенін білдірмейді.
Жабық әлем болжамы (Closed World Assumption) — белгілі бір жүйеде дәлелденбеген немесе сақталмаған факт жалған деп қабылдануы мүмкін.
Мысалы,
жабық тізімдегі тіркелген студенттерді тексергенде **тізімде жоқ** студент тіркелмеген деп есептелуі мүмкін.
Ал **ашық әлем тәсілінде тізімде** жоқ болуы оның тіркелмегенін дәлелдемейді. Ақпарат толық емес болуы мүмкін.
Бұл айырмашылық логикалық жүйелердің дұрыс жұмыс істеуі үшін маңызды.
18. Монотонды және монотонды емес логикалық шығару
Енді тағы бір маңызды айырмашылықты қарастырайық.
Монотонды логикалық шығару кезінде жаңа ақпарат қосылғанда бұрын логикалық түрде дәлелденген қорытындылар жойылмайды.
Ал монотонды емес логикалық шығару кезінде жаңа ақпарат бұрынғы қорытындыны қайта қарауға мәжбүр етуі мүмкін.
Мысал
Бізде мынадай әдепкі ереже бар делік:
Әдетте тіркелген студенттер университет кітапханасын пайдалана алады.
Студент **Айдана** тіркелген.
Жүйе бастапқыда:
«Айдана кітапхананы пайдалана алады»
деп қорытынды жасайды.

Кейін жаңа факт енгізіледі:
«Айдананың кітапханаға кіру құқығы уақытша тоқтатылған».
Енді бұрынғы қорытынды қайта қаралады.
Бұл — **монотонды емес** ой қорытуға тән жағдай.
Мұндай тәсілдер нақты өмірдегі ерекшеліктерді және әдепкі ережелерді өңдеуге мүмкіндік береді.

19. Логикалық шығару нәтижесін түсіндіру

Логикалық жүйе тек дұрыс қорытынды шығарумен **шектелмеуі** керек.
Көптеген жағдайларда оның қорытындыға қалай келгенін көрсету маңызды.
Мысалы, интеллектуалды жүйе:
«Серверге қосымша ресурс қажет»
деді делік.
Пайдаланушы бұл ұсыныстың себебін білгісі келеді.
Жүйе келесі тізбекті көрсете алады:
CPU = 94% және ResponseTime = 850 мс.
↓
R1 ережесі іске қосылды.
↓
ServerLoad = Critical.
↓
Белсенді пайдаланушылар саны = 1200.
↓
R2 ережесі іске қосылды.
↓
ScalingNeeded = Yes.
Бұл түсіндіру пайдаланушыға шешімнің негізін тексеруге мүмкіндік береді.
Сондықтан **логикалық шығару жүйелерінде** дәлелдеу тізбегі, **ереже журналы** және **қорытындының шығу тарихы** маңызды рөл атқарады.

20. Логикалық шығару механизмдерінің қолданылу салалары

Логикалық шығару механизмдері әртүрлі интеллектуалды жүйелерде қолданылады.

Сала	**Қолданылуы**
Сараптамалық жүйелер	Диагностика және ұсыныс беру
Білім беру	Оқу талаптарын тексеру, пәндер арасындағы байланыстар
Ақпараттық қауіпсіздік	Қолжетімділік ережелерін тексеру
Өндіріс	Жабдық жағдайын бағалау
Логикалық программалау	Мақсаттарды дәлелдеу
Білім графтары	Жаңа байланыстарды анықтау
Автоматтандырылған жоспарлау	Әрекеттердің орындалу шарттарын тексеру

Мысалы, ақпараттық қауіпсіздік жүйесінде:
Егер пайдаланушы әкімші болса және оның тіркелгісі белсенді болса, онда белгілі бір әкімшілік операцияға рұқсат беріледі.
деген ереже болуы мүмкін.
Бірақ жүйе қосымша шектеулерді де тексеруі керек:
операцияның түрі
пайдаланушының ағымдағы сессиясы
қауіпсіздік саясаты және басқа шарттар.
Сондықтан логикалық шығару нақты ережелер мен олардың арасындағы байланыстарды жүйелі түрде өңдеуге мүмкіндік береді.

21. Кешенді мысал: интеллектуалды қолжетімділікті басқару жүйесі

Енді дәрісте қарастырылған **негізгі механизмдерді** бір есепке біріктірейік.

Университетте ғылыми зертханаға қолжетімділікті басқаратын интеллектуалды жүйе жасалып жатыр делік.
Жүйе студенттің зертханаға кіруіне рұқсат беру немесе қосымша тексеру қажеттігін анықтайды.

Бастапқы фактілер

Көрсеткіш	**Мән**
Пайдаланушы	Дамир
Мәртебе	Магистрант
Қауіпсіздік нұсқаулығы	Өткен
Ғылыми жобаға тіркелуі	Бар
Зертханаға кіру уақыты	19:30
Рұқсат етілген қалыпты уақыт	09:00–18:00
Арнайы кешкі рұқсат	Жоқ

Білім базасындағы ережелер

R1: ЕГЕР студент қауіпсіздік нұсқаулығынан өткен ЖӘНЕ ғылыми жобаға тіркелген БОЛСА, ОНДА BasicAccess = Yes.

R2: ЕГЕР BasicAccess = Yes ЖӘНЕ кіру уақыты қалыпты уақыт аралығында болса, ОНДА AccessDecision = Allow.

R3: ЕГЕР BasicAccess = Yes ЖӘНЕ кіру уақыты қалыпты уақыттан тыс болса, ОНДА ExtendedAccessCheck = Required. (Кеңейтілген рұқсатты тексеру)

R4: ЕГЕР ExtendedAccessCheck = Required ЖӘНЕ арнайы рұқсат бар болса, ОНДА AccessDecision = Allow.

R5: ЕГЕР ExtendedAccessCheck = Required ЖӘНЕ арнайы рұқсат жоқ болса, ОНДА AccessDecision = Deny.

Шешім шығару

Бірінші кезеңде **R1** ережесінің шарттары орындалады:

- қауіпсіздік нұсқаулығынан өткен;
- ғылыми жобаға тіркелген.

Сондықтан:

BasicAccess = Yes

Екінші кезеңде жүйе кіру уақытын тексереді.

19:30 — қалыпты жұмыс уақытынан тыс.

Сондықтан **R2** орындалмайды, ал **R3** іске қосылады:

ExtendedAccessCheck = Required

Үшінші кезеңде жүйе арнайы рұқсатты тексереді.

Арнайы рұқсат жоқ.

Сондықтан **R5** іске қосылады:

AccessDecision = Deny

Нәтиже

Жүйе Дамирдің зертханаға осы уақытта кіруіне рұқсат бермейді.

Бірақ бұл оның зертханаға **мүлдем кіруге құқығы жоқ** дегенді білдірмейді.

Ол негізгі талаптарды орындаған. Тек **кешкі** уақытта кіру үшін қосымша рұқсат қажет.

Осы мысалда біз бірнеше маңызды нәрсені көрдік.

Біріншіден, бастапқы фактілерден аралық қорытынды алынды.

Екіншіден, уақытқа байланысты қосымша ереже іске қосылды.

Үшіншіден, қорытынды бірнеше шартты бірге тексеру арқылы шығарылды.

Төртіншіден, жүйе өз шешімін түсіндіре алады.

Бұл — логикалық шығару механизмдерінің нақты ақпараттық жүйелердегі қолданылуына мысал.

22. Логикалық шығару механизмдерінің артықшылықтары мен шектеулері

Логикалық шығару механизмдерінің **бірнеше** маңызды артықшылығы бар.

Олар білімді **формалды түрде** көрсетуге, қорытындыны **нақты ережелер** арқылы шығаруға, **шешімнің дәлелдеу** жолын бақылауға және **белгілі талаптарды автоматты** түрде тексеруге мүмкіндік береді.

Бірақ шектеулері де бар.

Егер білім базасы толық болмаса, жүйе қажетті қорытындыны шығара алмауы мүмкін.

Егер ережелер қате болса, қорытынды да қате болуы мүмкін.

Егер ережелер саны өте көп болса, оларды тиімді іздеу және орындау күрделене түседі.

Сонымен қатар классикалық логикалық жүйелер **белгісіздік**, **ықтималдық** және **толық емес** деректер сияқты жағдайларды өңдеу үшін қосымша әдістерді қажет етуі мүмкін.

Артықшылық	**Шектеу**
Формалды және тексерілетін қорытынды	Білім базасының сапасына тәуелді
Шешімді түсіндіруге болады	Ережелер саны өскен сайын күрделенеді
Нақты логикалық талаптарды өңдейді	Толық емес ақпаратпен жұмыс қиын болуы мүмкін
Автоматтандырылған дәлелдеу мүмкіндігі	Барлық нақты жағдайды ережемен сипаттау қиын

23. Логикалық шығару және машиналық оқыту арасындағы байланыс

Енді келесі дәріске дайындық ретінде бір маңызды мәселені қарастырайық.

Бүгін біз білім базасында алдын ала анықталған ережелер арқылы қорытынды шығаруды зерттедік.

Мысалы:

Егер CPU жүктемесі 90%-дан жоғары және жауап беру уақыты 700 миллисекундтан жоғары болса, **ОНДА** сервер жүктемесі критикалық деп бағаланады.

Бұл ережені адам анықтады.

Ал енді бізде бірнеше жыл бойы жиналған сервер жұмысының **үлкен деректер** жиынтығы бар делік.

Онда CPU жүктемесі, жедел жады, жауап беру уақыты, пайдаланушылар саны және сервердің істен шығу жағдайлары сақталған.

Біз осы деректерді пайдаланып, компьютердің өзіне белгілі заңдылықтарды анықтатқымыз келеді.

Бұл жерде **машиналық оқыту әдістері** қажет болады.

Логикалық шығару мен **машиналық оқыту** арасындағы негізгі айырмашылық:

Логикалық шығару — алдын ала берілген білімді қолданады.

Машиналық оқыту — деректерден заңдылықтарды үйренеді.

Бірақ бұл **екі** тәсілді біріктіруге болады.

Мысалы,

машиналық оқыту моделі сервердің істен шығу тәуекелін болжайды. Ал логикалық ережелер осы болжамға және қауіпсіздік талаптарына сүйеніп, қандай әрекетке рұқсат берілетінін анықтайды.

Осылайша біз келесі дәрістің тақырыбына келеміз.

24. Қорытынды

Сонымен, бүгін біз **логикалық шығару механизмдерін** тереңірек қарастырдық.

Ең алдымен логикалық шығару механизмінің интеллектуалды жүйеде қандай қызмет атқаратынын түсіндік.

Ол білім базасындағы фактілер мен ережелерді пайдаланып, жаңа тұжырымдар алуға немесе белгілі мақсатты дәлелдеуге мүмкіндік береді.

Біз негізгі логикалық ережелермен таныстық:

- Modus Ponens;
- Modus Tollens;
- тура логикалық шығару;
- кері логикалық шығару;
- унификация;

- резолюция.

Сонымен қатар предикаттар логикасының жалпы ережелерді сипаттауға мүмкіндік беретінін көрдік.

Ережелер арасында қайшылық пайда болуы мүмкін екенін және оларды шешу үшін басымдық, нақтылық және басқа стратегиялар қолданылатынын талдадық.

Толық емес ақпаратпен жұмыс істеу кезінде ашық және жабық әлем болжамдарының айырмашылығын түсіндік.

Монотонды және монотонды емес қорытынды шығарудың ерекшеліктерін қарастырдық.

Ең маңыздысы — **логикалық шығару механизмінің** мақсаты жай ғана жауап беру емес, берілген білім негізінде логикалық тұрғыдан негізделген қорытынды алу екенін түсіндік.

Ал келесі дәрісте біз **интеллектуалды жүйелердің** басқа бағытына — машиналық оқытудың негіздеріне көшеміз. Онда компьютердің дайын ережелерді ғана орындамай, деректерден заңдылықтарды қалай үйренетінін зерттейміз.

25. Бақылау сұрақтары

1. Логикалық шығару механизмі дегеніміз не?
2. Логикалық шығару мен арифметикалық есептеудің айырмашылығы қандай?
3. Логикалық қорытындының негізгі элементтерін атаңыз.
4. Modus Ponens ережесі қалай жұмыс істейді?
5. Modus Tollens ережесінің ерекшелігі қандай?
6. Қате логикалық қорытындының қандай түрлерін білесіз?
7. Тура логикалық шығару алгоритмі қандай кезеңдерден тұрады?
8. Кері логикалық шығару қалай жұмыс істейді?
9. Тура және кері шығару механизмдерін қандай жағдайларда қолданған тиімді?
10. Предикаттар логикасы не үшін қажет?
11. Унификация дегеніміз не?
12. Резолюция әдісінің негізгі идеясы қандай?
13. Хорн клаузалары дегеніміз не?
14. Prolog тілінде фактілер мен ережелер қалай жазылады?
15. Ережелер арасындағы қайшылықтар қалай шешіледі?
16. Логикалық шығару циклдері қандай мәселе туғызады?
17. Ашық әлем және жабық әлем болжамдарының айырмашылығы қандай?
18. Монотонды логикалық шығару дегеніміз не?
19. Монотонды емес логикалық шығару қандай жағдайда қажет?
20. Логикалық шығару нәтижесін түсіндіру не үшін маңызды?
21. Логикалық шығару механизмдері қай салаларда қолданылады?
22. Логикалық шығару мен машиналық оқытудың негізгі айырмашылығы қандай?

26. Сабақ соңындағы шағын тапсырма

Тақырыбы: Университеттің ғылыми зертханасына қолжетімділікті басқаратын логикалық жүйе

Университетте зертхана ресурстарын пайдалануға рұқсат беретін интеллектуалды жүйе жасалып жатыр.

Жүйеде келесі мәліметтер бар:

Көрсеткіш	**Мән**
Пайдаланушы	Әлихан
Мәртебе	Докторант
Қауіпсіздік нұсқаулығы	Өткен
Жобаға қатысуы	Расталған
Зертханаға кіру уақыты	20:15
Арнайы кешкі рұқсат	Бар
Құрылғыны пайдалануға рұқсат	Жоқ
Зертхана мәртебесі	Ашық

Тапсырма:

1. Берілген мәліметтерді логикалық фактілер түрінде көрсетіңіз.
2. Зертханаға кіруге және құрылғыны пайдалануға қатысты кемінде төрт өнімдік ереже құрыңыз.
3. Тура логикалық шығару әдісімен қандай қорытындылар алынатынын көрсетіңіз.
4. «Әлихан зертханадағы арнайы құрылғыны пайдалана ала ма?» деген мақсатты кері логикалық шығару әдісімен тексеріңіз.
5. Зертханаға кіру құқығы мен құрылғыны пайдалану құқығының айырмашылығын түсіндіріңіз.
6. Жүйенің соңғы шешімін және оған негіз болған ережелерді жазыңыз.

Бұл тапсырмада студент бір ғана фактіні тексеріп қоймай, бірнеше логикалық шарттың өзара байланысын талдауы қажет.