COURSE : FUNDAMENTALS OF COMPUTER SCIENCE

LECTURE 12: "ARTIFICIAL INTELLIGENCE"

Instructor:
PhD, Associate Professor
Tuyatsetseg Badarch

Зорилго

- Хиймэл оюун ухааныг тодорхойлж, товч түүхийг өгүүлэх.
- Мэдлэгийг intelligent agent-аар хэрхэн дүрсэлдэгийг тодорхойлох.
- Хүний ажиллагаагаар боломжгүй үед хэрхэн системийг ажиглаж болохыг харуулах.
- Хүний хийдэг энгийн ажлуудыг дуурайхад хиймэл оюуныг хэрхэн ашиглаж болохыг харуулах.
- Expert systems болон mundane systems-үүдээр асуудлыг шийдвэрлэхийн тулд янз бүрийн хайлтын арга техникийг хэрхэн ашиглаж болохыг харуулах.
- Перцептрон гэж нэрлэгддэг нейроны цахилгаан хэлбэрийг үүсгэдэг мэдрэлийн сүлжээг ашиглан хүний сургах үйл явцыг тодорхой хэмжээгээр хэрхэн загварчлах боломжтойг харуулах.

УДИРТГАЛ : Хиймэл оюун ухаан гэж юу вэ?

Хиймэл оюун ухаан гэж юу вэ?

Хиймэл оюун ухааны талаар олон нийтэд адилхан хүлээн зөвшөөрөгдсөн тодорхойлолт байдаггүй ч бид энэ бүлэгт хамрагдсан сэдвүүдэд хамаарах дараах байдлаар тодорхойлж болно.

Хиймэл оюун ухаан гэдэг нь ойлгох, сэтгэх, суралцах, үйлдэх зэрэг хүний үйл ажиллагааг тодорхой хэмжээгээр дуурайлгах боломжтой програмчлагдсан системүүдийн судалгаа юм.

Хиймэл оюун ухаан нь бие даасан харьцангуй шинэ судалгааны талбар боловч эрт дээр үеэс улбаатай. Энэ нь 2400 жилийн тэртээ Грекийн гүн ухаантан Аристотель логик сэтгэхүйн тухай ойлголтыг бий болгосноор эхэлсэн гэж хэлж болно. Түүнээс хойш Лейбниц, Ньютон нартай үргэлжилж Жорж Буль XIX зуунд Булийн алгебрийг боловсруулсан нь компьютерийн хэлхээний үндэс суурийг тавьсан юм. Ингэж явсаар хиймэл оюун ухаан гэдэг бол Тьюрингийн тестийг санаачилсан Алан Тюрингээс гаралтай. "Хиймэл оюун ухаан" гэсэн нэр томъёог анх 1956 онд Жон Маккарти гаргажээ.

УДИРТГАЛ : Хиймэл оюун ухаан гэж юу вэ?

Intelligent agent

Intelligent agent гэдэг нь орчноо ойлгож, түүнээс суралцаж, ухаалгаар харилцдаг тогтолцоо юм. Intelligent agent-ыг хоёр ангилалд хувааж болно: Software agents болон Physical agents.

Software agents

Sotware agent гэдэг нь тодорхой ажил үүргийг гүйцэтгэхэд зориулагдсан багц програм юм. Жишээ нь зарим ухаалаг системийг цахим шууданг (email) системчлэхэд ашиглаж болно. Энэ төрлийн agent нь хүлээн авсан имэйлийн агуулгыг шалгаж, өөр өөр ангилалд (хог, чухал биш, маш чухал, маш чухал гэх мэт) ангилж болно.

Physical agents

Physical agent нь олон төрлийн ажлыг гүйцэтгэхэд ашиглаж болох програмчлагдсан систем юм. Энгийн роботуудыг угсрах, гагнуур хийх зэрэг ердийн үйлдвэрлэлд ашиглаж болно. Зарим байгууллагууд өөр өөр өрөөнд шуудан, захидал илгээх зэрэг хүргэлтын ажилд хөдөлгөөнт роботуудыг ашигладаг. Усан доор газрын тос хайхад ашигладаг хөдөлгөөнт роботууд байдаг.

УДИРТГАЛ : Хиймэл оюун ухаан гэж юу вэ?

Programming languages

Хэдийгээр C, C++, Java зэрэг хэлийг intelligent software бүтээхэд ашигладаг ч LISP болон PROLOG гэсэн хоёр хэлийг тусгайлан хиймэл оюун ухаанд зориулан бүтээсэн.

LISP: LISP (LISt Programming) хэлийг 1958 онд Жон МакКарти зохион бүтээжээ. Нэрнээс нь харахад LISP нь жагсаалтыг удирдах програмчлалын хэл юм. LISP нь өгөгдөл болон программуудыг жагсаалт гэж үздэг бөгөөд энэ нь LISP програм нь өөрөө өөрийгөө өөрчлөх боломжтой гэсэн үг. Энэ онцлог нь хүрээлэн орчноосоо суралцаж, зан төлөвөө сайжруулж чаддаг intelligent agent-ыг илэрхийлдэг. Гэсэн хэдий ч LISP-ийн нэг сул тал бол түүний удаашрал юм. Хэрэв жагсаалт урт бол ажиллагаа удаан байна. Өөр нэг дутагдал нь түүний синтаксийн нарийн төвөгтэй байдал юм.

PROLOG: PROLOG (LOGic-д програмчлал) нь баримтуудын мэдээллийн сан(fact database), дүрмийн мэдлэгийн санг бүрдүүлж чаддаг хэл юм. PROLOG программ нь мэдлэгийн сангаас дүгнэлт хийж болох асуултуудад хариулахын тулд логик үндэслэлийг ашигладаг. Гэсэн хэдий ч PROLOG нь тийм ч үр дүнтэй програмчлалын хэл биш юм.

Мэдлэгийн илэрхийлэл

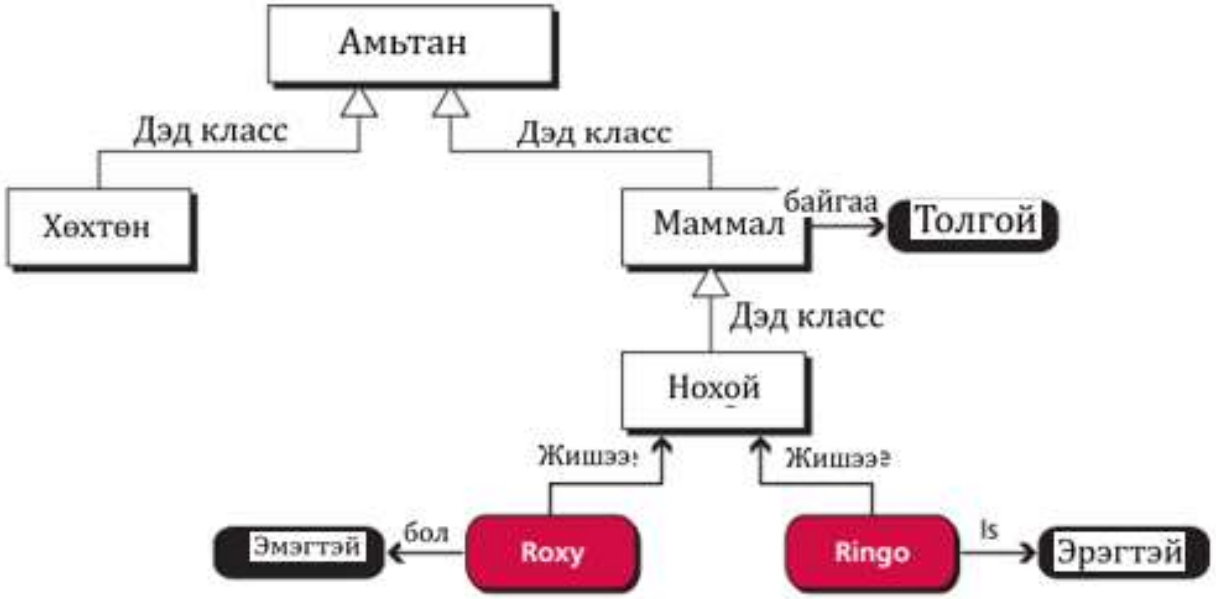
KNOWLEDGE REPRESENTATION

Хэрэв хиймэл agent бодит ертөнцтэй холбоотой аливаа асуудлыг шийдэх ёстой бол мэдлэг гэх зүйлийг ямар нэгэн байдлаар илэрхийлэх чадвартай байх ёстой. Баримтууд компьютерт программуудаар удирдаж болох өгөгдлийн бүтэц хэлбэрээр хадгалагддаг. Мэдлэгийг илэрхийлэх дөрвөн нийтлэг арга: *семантик сүлжээ*, *фрейм*, *предикатын логик*, *дүрэмд суурилсан систем*.

Semantic network

1960-аад оны эхээр Ричард Х.Риченс боловсруулсан. Семантик сүлжээ нь мэдлэгийг илэрхийлэхийн тулд directed graphic ашигладаг. Энэ нь directed graphic нь орой (зангилаа) ба ирмэгүүд (нум) -аас бүрдэнэ. Сүлжээ нь concept-ыг илэрхийлэхийн тулд оройг, хоёр concept хоорондын хамаарлыг илэрхийлэхийн тулд ирмэгийг (сумаар тэмдэглэсэн) ашигладаг.

Мэдлэгийн илэрхийлэл

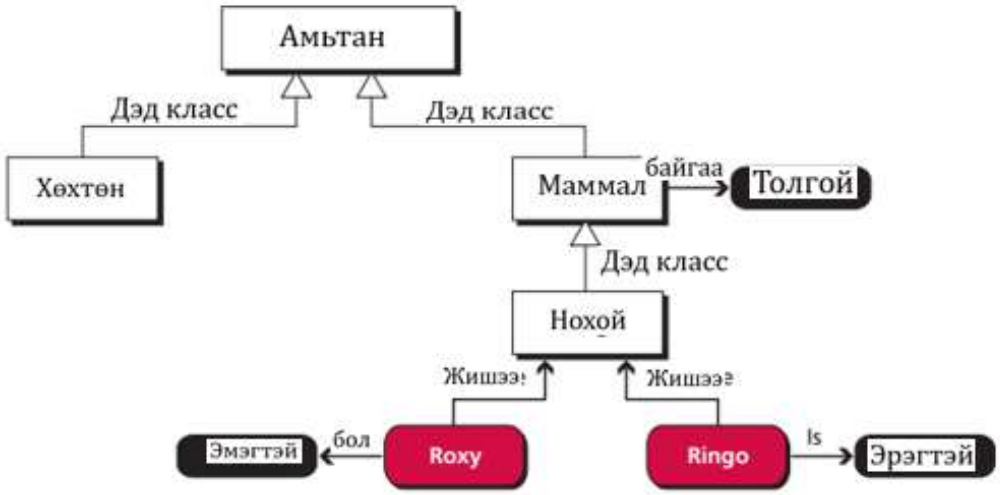


Зураг 1а. Семантик сүлжээний бүтэц

Concept

Concept-ыг тодорхойлохын тулд үүнийг олонлогтой холбож тайлбарлах шаардлага гарна. Тиймээс concept-ыг олонлог эсвэл дэд олонлог гэж үзэж болно. Жишээлбэл, амьтан бүх амьтны олонлог болж, морь нь бүх морыг төлөөлөхийн хамт амьтны дэд олонлог болно.. Объект нь олонлогийн гишүүн (жишээ) юм. Concept-ыг оройгоор харуулав.

Мэдлэгийн илэрхийлэл



Зураг 1б. Семантик сүлжээний бүтэц

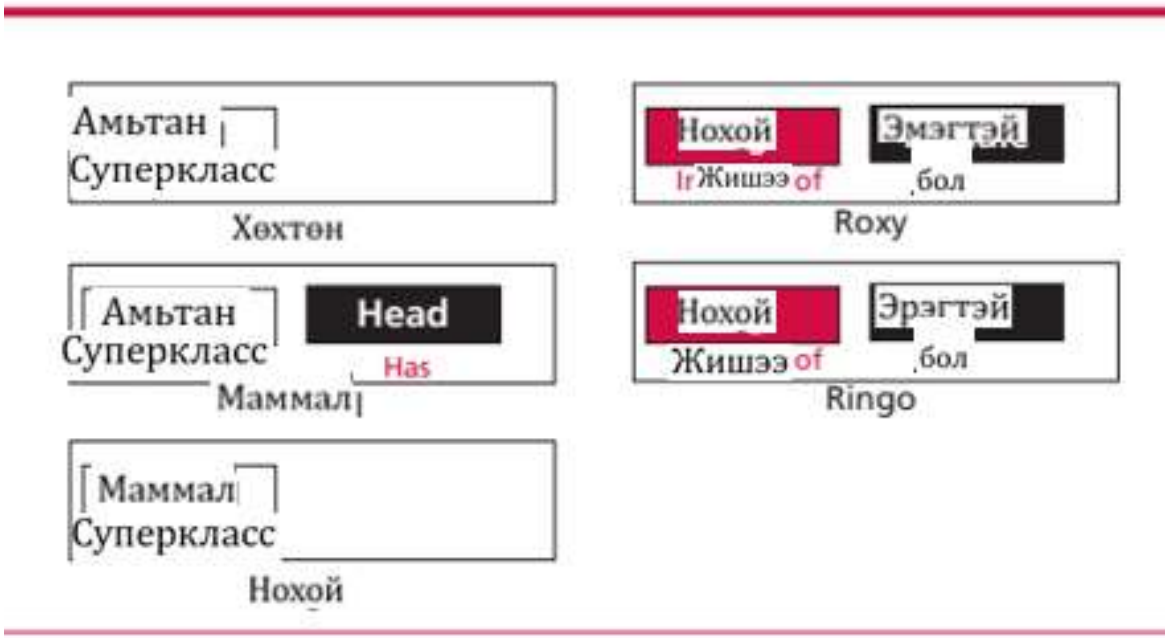
Relations

Семантик сүлжээнд хамаарлыг ирмэгээр харуулдаг. Ирмэг нь дэд классын хамаарлыг тодорхойлж чаддаг ба ирмэг нь дэд классаас эх класс руу гэсэн чиглэлтэй. Обьектыг мөн ирмэгээр дүрслэх боломжтой бөгөөд ирмэг нь объектын шинж чанарыг тодорхойлж болно (өнгө, хэмжээ,...). Семантик сүлжээнд сайн харагддаг хамгийн чухал хамаарлын нэг бол уламжлал юм. Уламжлалын хамаарал нь тухайн классын бүх шинж чанарууд нь уламжилсан классд байгааг харуулдаг. Үүнийг графикаар дүрсэлсэн мэдлэгээс шинэ мэдлэгийг гарган авахад ашиглаж болно.

Мэдлэгийн илэрхийлэл

Frames

Frames нь семантик сүлжээтэй нягт холбоотой байдаг. Семантик сүлжээнд мэдлэгийг илэрхийлэхэд графикийг ашигладаг. фреймд тэрхүү мэдлэгийг илэрхийлэхэд өгөгдлийн бүтцийг ашиглагддаг. Фреймийн нэг давуу тал нь программууд илүү фреймтэй ажиллах боломжтой байдаг. Зураг 1-д үзүүлсэн семантик сүлжээг фрейм ашиглан хэрхэн хэрэгжүүлж болохыг Зураг 2-т үзүүлэв.



Зураг 2. Семантик сүлжээг дүрслэх фрэймүүдийн олонлог

Мэдлэгийн илэрхийлэл

Objects

Семантик сүлжээн дэх node нь олон хүрээний объект болж хувирдаг тул объект нь анги, дэд анги эсвэл ангийн жишээг тодорхойлж чадна. Зураг 18.2-д хэвлээр явагч, хөхтөн, нохой, Рокси, Ринго зэрэг амьтад объектууд юм.

Slots

Семантик сүлжээн дэх ирмэгүүд нь slot буюу өгөгдлийн бүтцийн талбарууд руу хөрвүүлэгддэг. Слотын нэр нь классуудын хамаарлын төрөл болон хамаарлыг гүйцээж буй slot-ын утгыг тодорхойлно. Жишээлбэл, Зураг 18.2-т animal бол reptile объектын slot юм.

Мэдлэгийн илэрхийлэл

Predicate logic

Хамгийн түгээмэл knowledge representation бол Predicate logic. Энэ логикийг төвөгтэй баримтуудыг(facts) илэрхийлэхэд ашиглаж болно. Энэ бол онолын логикийг олон жилийн түүхээр дамжуулан боловсруулсан хэл юм. Predicate logic ийг үзэхээс өмнө арай энгийн хэл болох propositional logic-ыг судлая. Дараа нь бид propositional logic-ыг ашигладаг predicate logic-ын талаар ярилцана.

Propositional logic

Propositional logic гэдэг нь ертөнцийн талаархи логик үндэслэлийг хэрэгжүүлэхэд ашиглаж болох өгүүлбэрүүдийн багцаас бүрдсэн хэл юм.

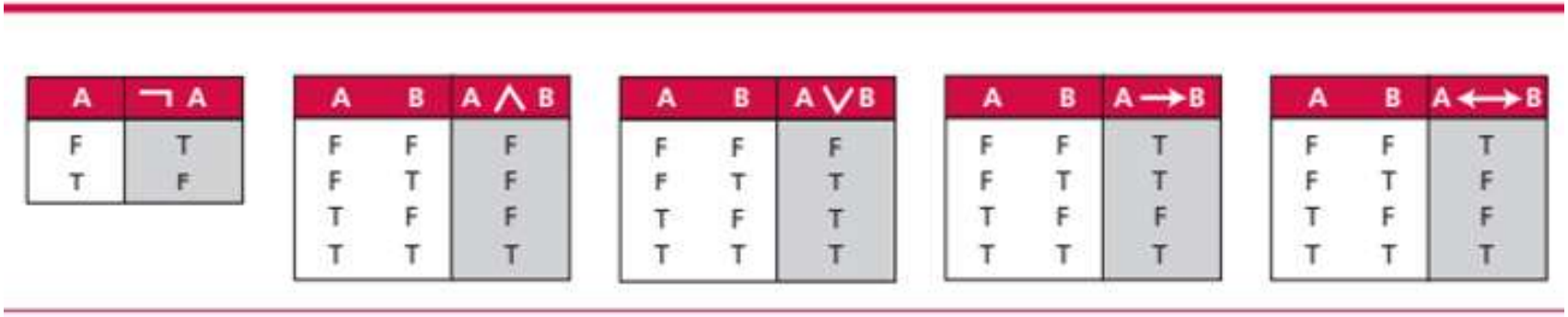
Operators

Propositional logic доор үзүүлсэн шиг таван оператор ашигладаг.

¬	∨	∧	→	↔
(not)	(or)	(and)	(if ... then)	(if and only if)

Мэдлэгийн илэрхийлэл

Эхний оператор нь unary-operator бөгөөд нэг өгүүлбэр авдаг. Бусад дөрвөн оператор нь хоёртын, хоёр өгүүлбэр авдаг. Өгүүлбэр бүрийн логик утга (үнэн эсвэл худал) нь complex өгүүлбэрийг бүрдүүлэх атомын өгүүлбэрүүдийн (операторгүй өгүүлбэр) логик утгаас хамаарна. Зураг 3-т propositional logic дахь логик оператор бүрийн үнэний хүснэгтийг үзүүлэв



A	¬A
F	T
T	F

A	B	A ∧ B
F	F	F
F	T	F
T	F	F
T	T	T

A	B	A ∨ B
F	F	F
F	T	T
T	F	T
T	T	T

A	B	A → B
F	F	T
F	T	T
T	F	F
T	T	T

A	B	A ↔ B
F	F	T
F	T	F
T	F	F
T	T	T

Зураг 3. Үнэний хүснэгт

Мэдлэгийн илэрхийлэл

Sentence (Өгүүлбэр)

Энэ хэл дээрх өгүүлбэрийг дараах байдлаар рекурсив байдлаар тодорхойлно.

1. Том үсэг, A, B, S гэх мэт эдгээр нь энгийн хэл дээрх өгүүлбэрүүдийг илэрхийлнэ.
2. Хоёр тогтмол утгын аль нэг нь (үнэн ба худал) өгүүлбэр юм.
3. Хэрвээ P өгүүлбэр мөн бол, $\neg P$ өгүүлбэр
4. Хэрвээ P болон Q өгүүлбэр бол, $P \vee Q$, $P \wedge Q$, $P \rightarrow Q$, $P \leftrightarrow Q$ нар өгүүбэр мөн.

Жишээ 18.1:

а. Өнөөдөр бол ням гараг(S).

б. Бороо орж байна (R).

в. Өнөөдөр Ням эсвэл Даваа гараг ($S \vee M$).

г. Энэ бороо орохгүй байна ($\neg R$)

д. Хэрвээ нохой бол хөхтөн амьтан бол муур хөхтөн хөхтөн амьтан($D \rightarrow C$)

Мэдлэгийн илэрхийлэл

Deduction(Дедукц)

AI-д бид одоо байгаа баримтуудаас шинэ баримтуудыг бий болгодог. Propositional logic-д энэ процессыг дедукц гэж нэрлэдэг. Үнэн байж магадгүй хоёр өгүүлбэрийг өгвөл бид шинэ үнэн өгүүлбэрийг гаргаж чадна. Эхний хоёр өгүүлбэрийг premisses(үндэслэл) гэж нэрлэдэг. Гаргаж авсан өгүүлбэрийг conclusion(дүгнэлт) гэж нэрлэдэг. Эдгээрыг бүхэлд нь аргумент гэнэ. Жишээлбэл:

Premiss 1:	Either he is at home or at the office
Premiss 2:	He is not at home
Conclusion	Therefore, he is at the office

Бид "Тэр гэртээ байгаа" гэсэн өгүүлбэрт H-г, "Тэр ажил дээрээ байгаа" гэсэнд O-г, дүнгэлтийг харуулахаар "|-" тэмбэгийг ашиглаж дээр дурдсан аргументийг дараах байдлаар харуулж болно:

$$\{H \vee O, \neg H\} \vdash O$$

Мэдлэгийн илэрхийлэл

Жишээ 18.2

{H ∨ O, ¬H} |- O аргументын үнэн зөвийг дараах үнэний хүснэгтийг ашиглан баталж болно.

H	O	H ∨ O	¬H	O	
F	F	F	T	F	
F	T	T	T	T	OK
T	F	T	F	F	
T	T	T	F	T	
		Прэмиз	Прэмиз	Дүгнэлт	

Шалгах цорын ганц мөр бол хоёр дахь мөр юм. Энэ мөрөнд counterexample байхгүй тул аргумент нь алдаагүй байна. Гэсэн хэдий ч логикийн хувьд хүчин төгөлдөр бус аргументууд байдаг. Жишээ:

Premiss 1:	If she is rich, she has a car.
Premiss 2:	She has a car
Conclusion	Therefore, she is rich.

Мэдлэгийн илэрхийлэл Жишээ

Эхний хоёр өгүүлбэр үнэн байсан ч дүгнэлт нь худал байж болохыг харж болно. Бид дээрх аргументыг {R → C, C} |– R гэж харуулж болох ба R нь 'Тэр баян', C нь 'тэр машинтай' гэсэн утгатай.

Жишээ 18.3

Counterexample олдож байгаа тул {R → C, C} |– R аргумент нь буруу .

R	C	R → C	C	R
F	F	T	F	F
F	T	T	T	F
T	F	F	F	T
T	T	T	T	T
		Прэмиз	Прэмиз	Дүгнэлт

Энд 2 дахь мөр болон 4 дахь мөр шалгах ёстой. 4 дахь мөр нь зөв байгаа боловч 2 дахь мөр нь хоёр үнэн premiss-ээс сөрөг дүгнэлт гаргаж байна. Тиймээс аргумент нь буруу байна.

Counterexample олдохгүй бол аргумент хүчинтэй байна.

Мэдлэгийн илэрхийлэл: атомт шинж чанар

Predicate logic

Predicate logic-д өгүүлбэрийг илэрхийлэх тэмдэг нь атомт шинж чанартай байдаг. Түүний бүрэлдэхүүн хэсгүүдийн талаархи мэдээллийг олохын тулд түүнийг задалж үзэх боломжгүй. Жишээлбэл, өгүүлбэрүүдийг авч үзье:

P_1: 'Linda is Mary's mother'	P_2: 'Mary is Anne's mother'

Бид энэ хоёр өгүүлбэрийг олон талаас нь нэгтгэн өөр өгүүлбэрийг бий болгож болох боловч Линда, Анн хоёрын хооронд ямар ч холбоог гаргаж чадахгүй. Жишээ нь, дээрх хоёр өгүүлбэрээс Линда бол Аннын эмээ гэж хэлж болохгүй. Ингэхийн тулд бидэнд Predicate logic хэрэгтэй. Таамаглалд буй хэсгүүдийн хоорондын уялдаа холбоог тодорхойлдог логик.

Predicate logic-т өгүүлбэрийг predicate, arguments болгон хуваан авч үздэг. Жишээлбэл, дараах өгүүлбэр бүрийг хоёр аргументтай predicate(предикат) хэлбэрээр бичиж болно

P_1: 'Linda is Mary's mother'	becomes	mother (Linda, Mary)
P_2: 'Mary is Anne's mother'	becomes	mother (Mary, Anne)

Мэдлэгийн илэрхийлэл: жишээ

Дээрх өгүүлбэр бүр дэх эх хүний харилцааг предикат эх тодорхойлдог. Хэрэв хоёр өгүүлбэрт байгаа Мэри нь нэг хүнийг хэлж байгаа бол бид Линда, Анне хоёрын хооронд шинэ харилцааг гаргаж болно. Эмээ (Линда, Энн). Энэ бол предикат логикийн зорилго юм.

Sentence (Өгүүлбэр)

Predicate language дээрх өгүүлбэрийг дараах байдлаар тодорхойлно.

1. Predicate_name (аргумент1,...аргументn) зэрэг n аргументтай предикат нь өгүүлбэр юм. Predicate_name нь аргументуудыг хооронд нь холбодог. Аргумент бүр нь:

   a. Хүн, амьтан, Жон, Мариа гэх мэт тогтмол.

   b. x, y, z гэх мэт хувьсагч.

   c. Ээж (Анне) гэх мэт функц. Функц нь аргумент болгон ашиглагддаг предикат гэдгийг анхаарна уу: функц нь аргументийн оронд байж болох объектыг буцаана.

2. Хоёр тогтмол утгын аль нэг нь (үнэн ба худал) өгүүлбэр юм.

3. Хэрвээ P өгүүлбэр мөн бол, ¬P өгүүлбэр

4. Хэрвээ P болон Q өгүүлбэр бол, $P \vee Q$, $P \wedge Q$, $P \rightarrow Q$, $P \leftrightarrow Q$ нар өгүүбэр мөн.

Мэдлэгийн илэрхийлэл : Тоон үзүүлэлт

Жишээ 18.4

а. "Жон Анны эгчид ажилладаг" өгүүлбэрийг дараах байдлаар бичиж болно. -> works [John, sister (Ann)]

Дээр эгч (Анн) функцийг аргумент болгон ашиглаж байна.

б. "Жоны аав Анны эгчид хайртай" гэсэн өгүүлбэрийг дараах байдлаар бичиж болно. -> loves [father (John), sister (Ann)]

Quantifiers

Предикатын логик нь тоон үзүүлэлтийг(**Quantifiers**) ашиглах боломжийг бидэнд олгодог. Нийтлэг байдаг хоёр quantifiers: ∀ ба ∃.

1."Бүх нийтийн"(‘for all’) гэж уншдаг эхний ∀-г бүх нийтийн хэмжигч(*universal quantifier*) гэж нэрлэдэг: энэ нь хувьсагчаар төлөөлж буй объект бүрийн хувьд ямар нэг зүйл үнэн болохыг илэрхийлдэг.

2. Хоёрдахь ∃ нь "байгаа"(‘there exists’) гэж уншигдах бөгөөд үүнийг оршихуйн хэмжигч(*existential quantifier*) гэж нэрлэдэг: энэ нь хувьсагчаар төлөөлж буй нэг буюу хэд хэдэн объектын хувьд ямар нэг зүйл үнэн болохыг илэрхийлдэг.

Мэдлэгийн илэрхийлэл : Жишээ

Дараах нь англи хэл дээрх өгүүлбэрүүдийг хэрхэн предикат логикоор өгүүлбэр болгон бичиж болохыг харуулна.
(X нь хөрвүүлэгдэхдээ):

1. "Бүх хүмүүс үхнэ" гэсэн өгүүлбэрийг дараах байдлаар бичиж болно.

$$\forall x\ [man\ (x) \rightarrow mortal\ (x)]$$

2. "Мэлхийнүүд ногоон" өгүүлбэрийг дараах байдлаар бичиж болно.

$$\forall x\ [frog\ (x) \rightarrow green\ (x)]$$

3. "Зарим цэцэг улаан" гэсэн өгүүлбэрийг дараах байдлаар бичиж болно.

$$\exists x\ [flower\ (x) \wedge red(x)]$$

Хаалт доторх оператор нь → биш харин ∧ гэдгийг анхаарна уу.

4. "Жон номтой" өгүүлбэрийг дараах байдлаар бичиж болно.

$$\exists\ x\ [book\ (x) \wedge has\ (John, x)$$

Өөрөөр хэлбэл, өгүүлбэрийг "Жонд харьяалагддаг ном оршдог" гэж өөрчилсөн.

5. "Ямар ч мэлхий шар биш"

$$\forall x[frog\ (x) \rightarrow \neg yellow\ (x)]\ \text{or as}\ \neg\exists x\ [frog\ (x) \wedge yellow\ (x)]$$

Мэдлэгийн илэрхийлэл : дедукц

Deduction(Дедукц)

Предикатын логикт quantifier байхгүй бол аргументыг баталгаажуулах нь бидний propositional logic дээр үзсэнтэй ижил байна. Гэсэн хэдий ч quantifiers байгаа тохиолдолд баталгаажуулалт илүү төвөгтэй болно. Жишээлбэл, дараах аргумент бүрэн хүчинтэй байна.

Premiss 1:	All men are mortals.
Premiss 2:	Socrates is a man.
Conclusion	Therefore, Socrates is mortal.

Энэхүү энгийн аргументыг баталгаажуулах нь тийм ч хэцүү биш юм. Бид энэ аргументыг дараах байдлаар бичиж болно.

$$\forall x\,[\text{man}(x) \rightarrow \text{mortal}(x)]\,,\ \text{man}(\text{Socrates}) \vdash \text{mortal}(\text{Socrates})$$

Мэдлэгийн илэрхийлэл : Аргумэнт

Эхний premises бүх хүнийг хэлж тул бид тухайн үндэслэл дэх хүн классын (Сократ)-ын нэг жишээг орлуулж дараах аргументыг гаргаж болно.

man (Socrates) → mortal (Socrates) , man (Socrates) |- mortal (Socrates)

M1 → M2 , M1 |− M2 болж багассан бөгөөд үүнд M1 нь хүн (Сократ), M2 нь мөнх бус (Сократ). Үр дүн нь propositional logic дахь аргумент бөгөөд үүнийг хялбархан батлах боломжтой. Гэсэн хэдий ч предикатын логикт тийм ч амархан батлагддаггүй олон аргументууд байдаг. Бидэнд энэ номын хамрах хүрээнээс давсан олон нотолгоо хэрэгтэй болно.

Beyond predicate logic

Логик үндэслэлийн хэрэгцээг оруулахын тулд логикийг цааш нь илүү хөгжүүлсэн. Эдгээрийн зарим жишээнд өндөр high-order logic, default logic, modal logic, temporal logic зэрэг орно. Эдгээр сэдвүүдийг зөвхөн сонирхлын үүднээс товчхон дурдаж байна.

Мэдлэгийн илэрхийлэл : Дүрэмд суурилсан систем

Rule-based systems

Дүрэмд суурилсан систем нь мэдэгдэж буй баримтаас шинэ баримт гаргахад ашиглаж болох дүрэм журмын багцыг ашиглан мэдлэгийг илэрхийлдэг. Дүрэм нь тодорхой нөхцөл хангагдсан тохиолдолд юу үнэн болохыг илэрхийлдэг. Дүрэмд суурилсан мэдээллийн сан нь if… then… хэллэгүүдийн багц юм.

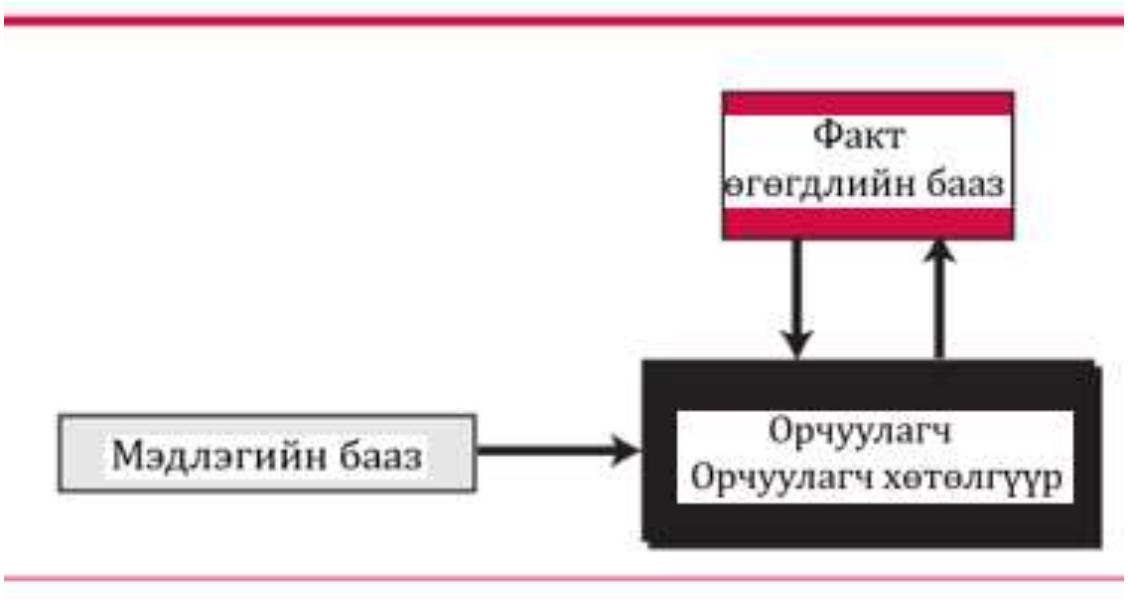
If A then B or $A \rightarrow B$

Үүнд А-г *antecedent*, Б-г *consequent*. гэж нэрлэдэг. Дүрэмд суурилсан системд дүрэм бүрийг бусад дүрэмтэй ямар ч холбоогүйгээр бие даан авч үздэг гэдгийг анхаарна уу.

Мэдлэгийн илэрхийлэл : бүрэлдэхүүн хэсэг

Components

Дүрэмд суурилсан систем нь 4-р зурагт үзүүлсэн шиг a interpreter (or inference engine), a knowledge base, a fact database гэсэн гурван бүрэлдэхүүн хэсгээс бүрдэнэ.



Зураг 4. Дүрэмд суурилсан системийн бүрэлдэхүүн хэсэг

Мэдлэгийн илэрхийлэл: repository

Knowledge base

Дүрэмд суурилсан систем дэх knowledge base бүрэлдэхүүн хэсэг нь дүрмийн database(repository) юм. Үүнд өгөгдсөн баримтаас дүгнэлт гаргахад ашиглаж болох урьдчилан тогтоосон дүрмүүд өгөгдсөн байдаг.

Database of facts

Database of facts нь knowledge base-ын дүрмүүдэд хэрэглэгддэг нөхцлүүдийн багцыг агуулдаг.

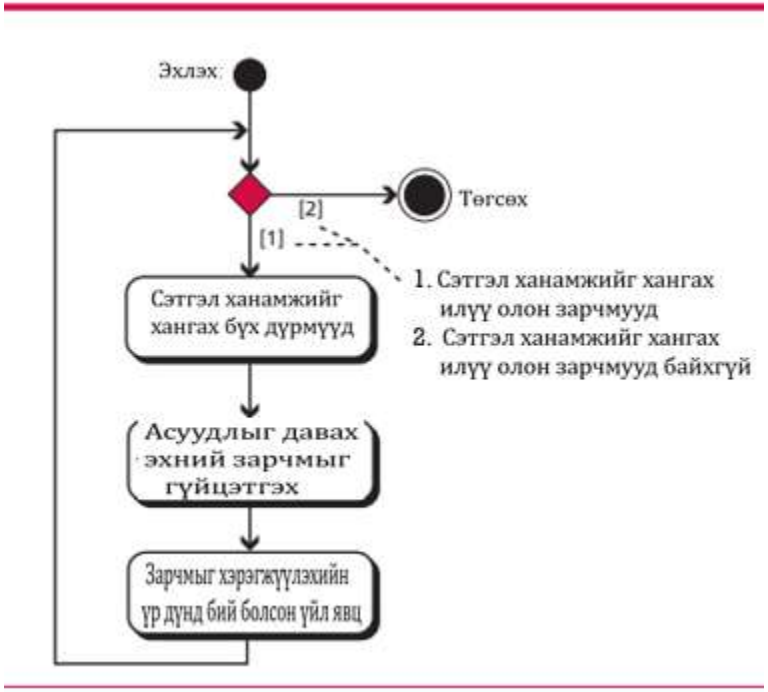
Interpreter

Interpreter (inference engine) нь дүрэм, баримтуудыг нэгтгэсэн процессор эсвэл удирдлага, программ юм. Interpreter нь *forward chaining* ба *backward chaining* гэсэн хоёр төрөлтэй.

Мэдлэгийн илэрхийлэл: forward chaining

Forward chaining

Forward chaining гэдэг нь interpreter тодорхой дүрэм журам, олон баримтыг ашиглан үйлдэл хийх үйл явц юм. Үйлдэл нь зөвхөн base of facts дээр шинэ баримт нэмэх, эсвэл өөр програм эхлүүлэх гэх мэт зарим тушаалуудыг өгөх явдал байж болно. Interpreter нь дүрмийг тайлбарлах боломжгүй болтол тайлбарлаж, гүйцэтгэдэг. Зураг 5-д үндсэн алгоритмыг үзүүлэв.

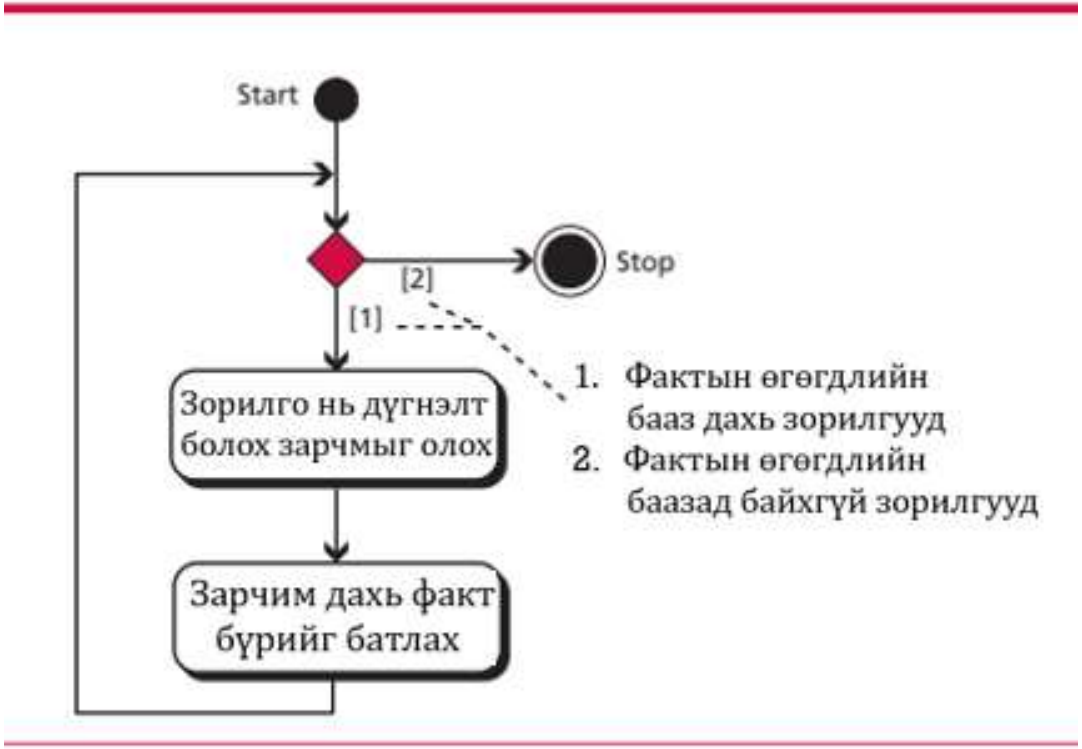


Зураг 5. Өгөгдлийн урсгалын диаграм

Мэдлэгийн илэрхийлэл: Backward chaining

Backward chaining

Хэрэв систем дүгнэлтийг батлах гэж оролдвол forward chaining нь тийм ч үр дүнтэй биш юм. Өгөгдсөн дүгнэлтийг гаргахын тулд бүх баримтыг бүх дүрмээр шалгаж үзэх ёстой. Энэ тохиолдолд backward cahining хэлхээг ашиглавал илүү үр дүнтэй байх болно. Зураг 6-д үндсэн алгоритмыг харуулав.



Зураг 6. Өгөгдлийн урсгалын диаграм , буцах урсгал

Эксперт систем

EXPERT SYSTEMS

Expert system-үүд нь өмнөх хэсэгт авч үзсэн мэдлэгийг илэрхийлэх хэлүүдийг ихэвчлэн хүний мэргэжил туршлага шаарддаг ажлыг гүйцэтгэхэд ашигладаг. Тэдгээрийг тухайн мэргэжлийн ур чадвар хомс, үнэтэй эсвэл шаардлагатай үед тохиолдолд ашиглаж болно. Жишээлбэл, анагаах ухаанд expert system нь шинж тэмдгүүдийн багцыг магадлалтай дэд бүлэгт хүргэж нарийсгаж чаддаг.

Extracting knowledge

Expert system нь тухайн мэргэжлийн чиглэлээр урьдчилан тодорхойлсон мэдлэг дээр суурилдаг. Жишээлбэл, анагаах ухааны expert system нь тухайн системийг бий болгох чиглэлээр мэргэшсэн эмчийн мэдлэг дээр суурилдаг: expert system нь эмчтэй ижил ажлыг гүйцэтгэх ёстой. Тиймээс expert system-ыг байгуулах эхний алхам бол хүнээс мэдлэг олж авах явдал юм. Энэхүү гаргаж авсан мэдлэг нь бидний өмнөх хэсэгт авч үзсэн knowledge base болдог.

Эксперт систем : мэргэжилтэн шиг ажилладаг машин

Мэргэжилтнээс мэдлэг олж авах нь хэд хэдэн шалтгааны улмаас хэцүү байдаг:
1. Мэргэжилтнүүдийн мэдлэг нь ихэвчлэн heuristic шинж чанартай байдаг: энэ нь тодорхой бус харин магадлал дээр суурилдаг.
2. Мэргэжилтнүүд мэдлэгээ тодорхой дүрмийн дагуу мэдлэгийн санд хадгалах боломжтой байдлаар илэрхийлэхэд хэцүү байдаг. Жишээлбэл, цахилгааны инженер ажилладаггүй цахилгаан моторыг хэрхэн оношлохыг алхам алхмаар харуулахад хэцүү. Мэдлэг нь ихэвчлэн мэдрэмжтэй холбоотой байдаг.
3. Мэдлэг олж авах нь зөвхөн мэргэжилтэнтэй хувийн ярилцлага хийх замаар л хийгдэх бөгөөд хэрэв ярилцлага авагч ийм төрлийн ярилцлагын мэргэжилтэн биш бол мэргэжилтэнд ядаргаатай, уйтгартай санагдаж болно.

Extracting facts

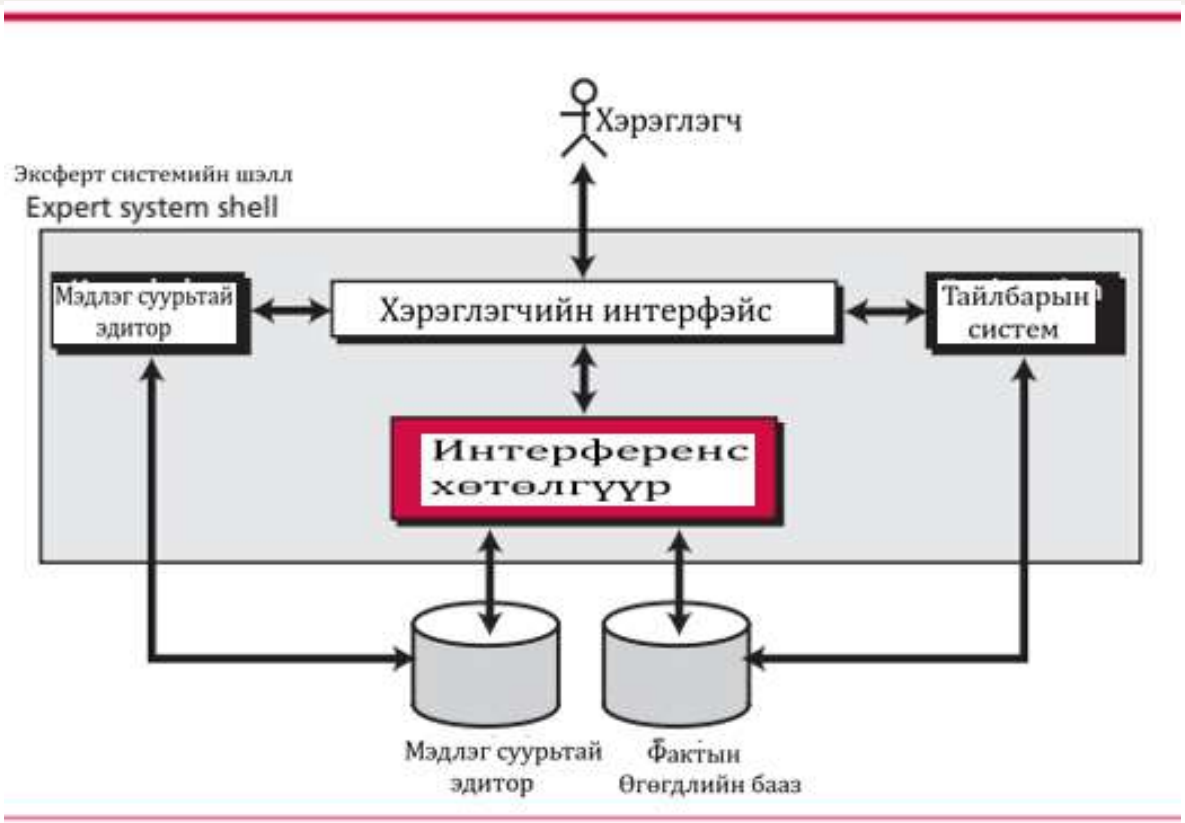
Шинэ баримт дүгнэлт гаргах эсвэл үйлдлийг гүйцэтгэхийн тулд мэдлэгийг илэрхийлэх хэлний knowledge base-ээс гадна fact database шаардлагатай. Expert system дэх fact database нь кейс дээр суурилсан бөгөөд цуглуулж хэмжсэн баримтуудыг inference engine-д ашиглахын тулд системд оруулдаг.

Эксперт систем бүтэц

Architecture

Inference engine нь Expert system-ын зүрх юм: энэ нь knowledge base, fact database, user interface-тай харилцдаг. Expert system-ын бүрэлдэхүүн user interface, inference engine, explanation system, knowledge base editor дөрөв нь тодорхой knowledge base эсвэл fact database-аас хамаардаггүй тул нэг удаа хийчхээд, олон хэрэглээнд ашиглаж болно. Зурагт эдгээр бүрэлдэхүүн хэсгүүдийг ихэвчлэн *expert system shell* гэж нэрлэдэг сүүдэртэй хайрцагт харуулав.

Эксперт систем бүтэц



Зураг 7. Эксперт системийн бүтэц

Зураг 7-д Expert system-ын архитектурын ерөнхий санааг харуулав. Зургаас харахад шинжээч систем нь *user, user interface, inference engine, knowledge base, fact database, explanation system, knowledge base editor* гэсэн долоон бүрэлдэхүүн хэсэгтэй байж болно.

Эксперт систем : жишээнүүд

User: User нь санал болгож буй мэдлэг, туршлагаас ашиг хүртэхийн тулд системийг ашигладаг байгууллага юм.
User interface: User interface нь хэрэглэгчийг системтэй харилцах боломжийг олгодог. User interface нь хэрэглэгчээс хүний хэлийг хүлээн авч, үүнийг системд хөрвүүлэх боломжтой.
Inference engine: Inference engine нь knowledge base, the fact database-ыг ашиглан гүйцэтгэх үйлдлийн талаар дүгнэлт гаргах системийн зүрх юм.
Knowledge base: Knowledge base нь тухайн мэргэжлийн чиглэлээр мэргэшсэн мэргэжилтнүүдтэй хийсэн ярилцлагад үндэслэсэн мэдлэгийн цуглуулга юм.
Fact database: Fact database нь кейс дээр суурилсан бөгөөд цуглуулж хэмжсэн баримтуудыг inference engine-д ашиглахын тулд системд оруулдаг.
Explanation system: Заавал байх албагүй Explanation system-ыг inference engine-ын гаргасан шийдвэрийн логик үндэслэлийг тайлбарлахад ашигладаг.
Knowledge base editor: Заавал байх албагүй тухайн салбарын мэргэжилтнүүдээс шинэ туршлага олж авсан тохиолдолд knowledge base-ыг шинэчлэхэд ашигладаг.

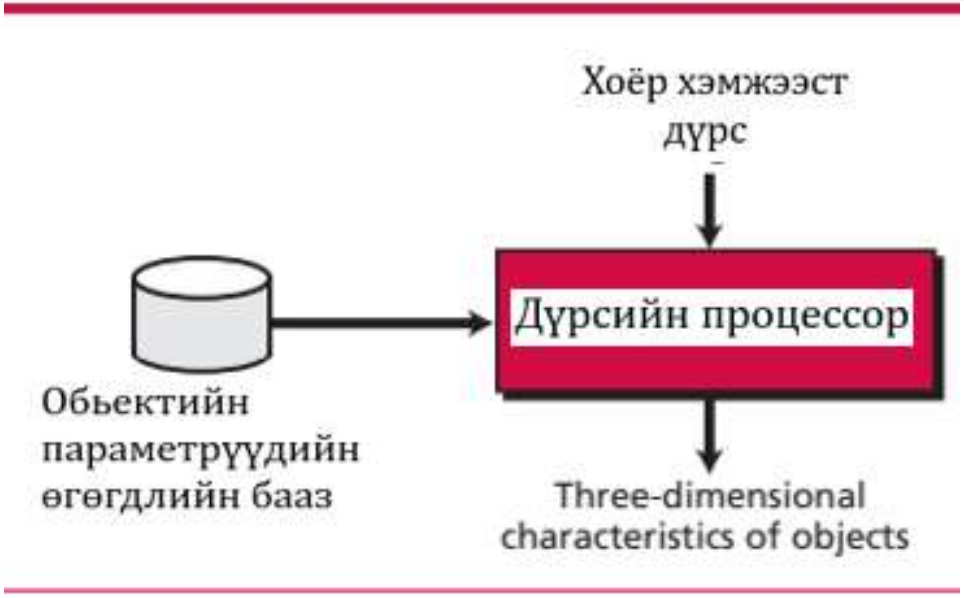
PERCEPTION : Мэдрэхүйгээр дамжуулах

PERCEPTION

Хиймэл оюун ухааны нэг зорилго бол мэргэжилтэн шиг ажилладаг машин буюу expert system-ыг бий болгох явдал юм. Өөр нэг зорилго бол энгийн хүн шиг ажилладаг машин бүтээх явдал юм. "Perception" гэдэг үгийн утга нь харах, сонсох, хүрэх, үнэрлэх, амтлах гэх мэт мэдрэхүйгээр дамжуулан хүлээн авсан зүйлийг ойлгох явдал юм. Хүн зургийг нүдээр харж, тархи нь тухайн зургийг бидэнд ойлгогдохоор хөрвүүлж байдаг. Хүн чихээрээ олон тооны дуут дохиог сонсож, тархи үүнийг утга учиртай өгүүлбэр болгон тайлбарлах гэх мэт.

Intelligent agent хүн шиг үйлдэхийн тулд ойлгож чаддаг байх ёстой. Ирээдүйд бусад төрлийн ойлголт хэрэгжиж болох хэдий ч AI нь хоёр төрлийн ойлголтод(хараа, сонсгол) оролцдог болсон. Энэ хэсэгт бид энэ хоёр зүйлийн талаар ярилцана.

PERCEPTION : дүрс боловсруулах



Зураг 8. Дүрсийн процессорын бүрэлдэхүүн хэсэг

Image processing

Дүрс боловсруулах буюу компьютерийн хараа нь камер гэх мэт төлөөлөгчийн хиймэл нүдээр объектыг хүлээн авах үйл ажиллагаа явуулдаг хиймэл оюун ухааны салбар юм. Дүрс боловсруулагч нь гадаад ертөнцөөс хоёр хэмжээст дүрсийг авч, тухайн үзэгдэлд буй гурван хэмжээст объектын зураглалыг бий болгохыг оролддог. Хэдийгээр энэ нь хүний хувьд амархан ажил боловч artificial agent хувьд хэцүү ажил болж хувирдаг. Зургийн процессорт үзүүлсэн оролт нь тухайн үзэгдлийн нэг буюу хэд хэдэн зураг, гаралт нь тухайн үзэгдэл дэх объектуудын зураглал гэсэн үг юм. Процессор нь харьцуулалд объектуудын шинж чанарыг агуулсан мэдээллийн санг ашигладаг (Зураг 8).

PERCEPTION : ирмэг илрүүлэлт

Зураг 9. Ирмэг илрүүлэлтийн процесс

Edge detection

Зургийн боловсруулалтын эхний үе шат бол ирмэгийг илрүүлэх явдал юм. Ирмэгүүд нь зураг дээрх объект болон түүний дэвсгэр хоорондын хил хязгаарыг тодорхойлж чадна. Ер нь бол ямар нэгэн биетэд хамаарах гадаргуу, орчин хоёрын хооронд эрс тэс ялгаа байдаг. Ирмэгүүд нь гадаргуу, гүн эсвэл гэрэлтүүлгийн тасалдлыг харуулдаг. Жишээлбэл, Зураг 9-д маш энгийн зураг, пикселийг 0-ээс 9 хүртэлх масштабаар харуулсан бөгөөд 0 нь хар, 9 нь цагаан байна. Зөрүү ихтэй зэргэлдээх пикселүүдийг олох замаар ирмэгийг илрүүлж болно.

PERCEPTION : сэнмэнтчлэл

Segmentation

Сегментчлэл нь зургийн боловсруулалтын дараагийн шат юм. Сегментаци нь зургийг нэгэн төрлийн сегмент эсвэл хэсгүүдэд хуваадаг. Нэгэн төрөл гэдэг нь ерөнхийдөө пикселийн нягтрал нь жигд өөр өөр байдаг хэсгүүд юм. Сегмент нь ирмэг илрүүлэхтэй маш төстэй. Ирмэгийг илрүүлэхэд объектын хил хязгаар, дэвсгэр нь олддог. Сегментчилэлд объектын доторх өөр өөр хэсгүүдийн хоорондох хил хязгаарыг олдог. Сегментчилсний дараа объектыг олон хэсэгт хуваана.

Сегментчилэхэд хэд хэдэн аргыг ашигладаг. Нэгийг нь *thresholding* гэж нэрлэдэг бөгөөд тодорхой нягтралтай пикселийг сонгож, ижил эсвэл ойролцоо нягтралтай бүх пикселийг олохыг оролддог. Ийм аргаар олдсон бүх пикселүүд нь сегмент үүсгэдэг. Өөр нэг аргыг splitting гэж нэрлэдэг. Splitting нь нэг төрлийн биш талбайг авч, хэд хэдэн нэг төрлийн хэсэгт хуваана. Өөр нэг аргыг нэгтгэх гэж нэрлэдэг бөгөөд үүнийг ижил пикселийн нятралтай хэсгүүдийг нэгтгэхэд ашиглаж болно.

PERCEPTION : Гүн тодорхойлох

Finding depth

Зургийн шинжилгээний дараагийн алхам бол зураг дээрх объектын гүнийг олох явдал юм. Гүн олох нь intelligent agent-д тухайн объект түүнээс хэр хол байгааг тодорхойлоход тусална. Энэ зорилгоор *stereo vision* болон *motion* гэсэн хоёр ерөнхий аргыг ашигладаг.

Stereo vision

Стерео хараа (заримдаа *stereopsis* гэж нэрлэдэг) нь объектын гүнийг олохын тулд хүний хараанд тулгуурлаж хөгжүүлсэн техникийг ашигладаг. Хоёр камераар бүтсэн зураг нь intelligent agent-д тухайн объект ойрхон эсвэл хол байгаа эсэхийг тодорхойлоход тусална.

Motion

Зурган дээрх объектуудын зайг олоход туслах өөр нэг арга бол нэг буюу хэд хэдэн объект хөдөлж байхад хэд хэдэн зураг авах явдал юм. Хөдөлгөөнт объектын дүр зурагийг бусад объекттой харьцуулахад гарах харьцангуй байрлал нь объектуудын зайг тодорхойлох боломж олгодог.

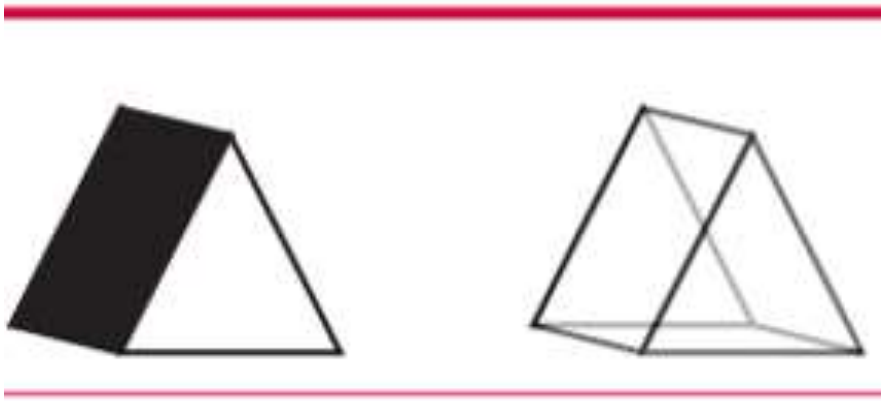
PERCEPTION : обьектын чиглэл

Finding Orientation

Үзэгдэл дээрх объектын чиглэлийг *shading*, *texture* гэсэн хоёр аргыг ашиглан олж болно.

Shading

Биетээс ойсон гэрлийн хэмжээ хэд хэдэн хүчин зүйлээс хамаарна. Хэрэв объектын өөр өөр гадаргуугийн оптик шинж чанарууд ижил байвал тусгалын хэмжээ нь гэрлийн эх үүсвэрийг ойлгосон гадаргуугийн чиглэлээс (түүний харьцангуй байрлалаас) хамаарна. Зураг 10-д хоёр объектыг үзүүлэв. Сүүдэрлэсэн нь объектын гадаргуугийн чиглэлийг илүү нарийвчлалтай харуулж байна.



Зураг 10. Сүүдрийн эффект

PERCEPTION : гадаргуу

Texture

Бүтэц (байнга давтагддаг загвар) нь гадаргуугийн чиг баримжаа эсвэл муруйлтыг олоход тусалдаг. Intelligent agent бүтцийг тодорхойлж чадвал объектын чиг баримжаа эсвэл муруйлтыг олж болно.

Object recognition

Зураг боловсруулах сүүлийн алхам бол объектыг таних явдал юм. Объектыг танихын тулд agent нь санах ойд харьцуулах объектын загвартай байх шаардлагатай. Гэсэн хэдий ч, харагдаж буй объект бүрийн загвар үүсгэж, хадгалах нь боломжгүй ажил юм. Нэг шийдэл бол объектуудыг энгийн геометрийн дүрсүүдээс бүрдсэн нийлмэл биет гэж үзэх явдал юм. Эдгээр энгийн дүрсүүдийг үүсгэж, agent-ын санах ойд хадгалж, дараа нь эдгээр объектуудын нийлж үүсгэх шаардлагатай объектын классуудыг үүсгэж, хадгалж болно. Агент объектыг "харах" үедээ уг объектыг дүрсын хослол болгон задлахыг оролддог. Хэрэв хослол нь тухайн объектод аль хэдийн мэдэгдэж байсан классын аль нэгтэй тохирч байвал тухайн объект танигдана. Зураг 18.11-д анхдагч геометрийн дүрсүүдийн жижиг багцыг үзүүлэв.

PERCEPTION : хэрэглээнүүд

Зураг 11. Геометрийн дүрсүүд

Applications

Зураг боловсруулах нь үйлдвэрлэлд, ялангуяа угсралтын салбарт хэрэглэгдэж байна. Зураг 11- д боловсруулах чадвартай роботыг угсрах шугам дээрх объектын байрлалыг тодорхойлоход ашиглаж болно. Хүлээн авах объектын тоо хязгаарлагдмал энэ орчинд зураг боловсруулагч маш их тустай.

PERCEPTION : эх хэлний асуудлууд

Language understanding

Хүний төрөлхийн өөр чадваруудын нэг бол хүлээн авч буй аудио дохиог ойлгох явдал юм. Хүний хэлийг ойлгох чадвартай машин нь өдөр тутмын амьдралд их үр дүнтэй байж болно.

Хүний хэлийг ойлгодог машины үйл ажиллагааг бид *speech recognition*, *syntactic analysis*, *semantic analysis*, *pragmatic analysis* гэсэн дөрвөн дараалсан үе шатанд хуваааж болно.

Speech recognition

Энэ үе шатанд ярианы дохиог шинжилж, түүнд агуулагдах үгсийн дарааллыг гаргаж авдаг. Speech recognition subsystem-ын оролт нь тасралтгүй (аналог) дохио юм: гаралт нь үгсийн дараалал юм.

Syntactic analysis

Синтактик анализын алхам нь үг өгүүлбэрт хэрхэн бүлэглэгдэхийг тодорхойлоход хэрэглэгддэг.

PERCEPTION : сэмантик анализ

Semantic analysis

semantic analysis нь өгүүлбэрийн синтаксыг шинжилсний дараа түүний утгыг гаргаж авахад хэрэглэгдэнэ. Энэ дүн шинжилгээ нь өгүүлбэрт хамаарах объектууд, тэдгээрийн харилцаа холбоо, шинж чанаруудын дүрслэлийг бий болгодог. Шинжилгээнд бидний өмнө үзсэн мэдлэгийг илэрхийлэх схемүүдийн аль нэгийг ашиглаж болно. Жишээлбэл: "Жон нохойтой" өгүүлбэрийг предикат логик ашиглан дараах байдлаар илэрхийлж болно.

$$\exists x \; \text{dog}(x) \; \text{has} \; (\text{John}, x)$$

Pragmatic analysis

Өмнөх гурван үе шат болох speech recognition, syntax analysis, semantic analysis зэрэг нь ярианы өгүүлбэрийн мэдлэгийн дүрслэлийг бий болгож чадна. Өгүүлбэрийн зорилгыг илүү тодорхой болгох, тодорхой бус байдлыг арилгахын тулд өөр нэг алхам болох прагматик дүн шинжилгээ хийх шаардлагатай байдаг.

PERCEPTION : зорилго тодорхойлох

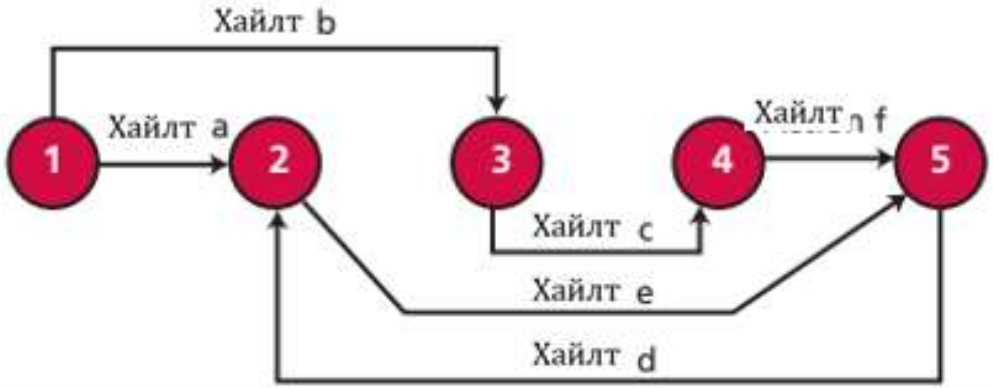
Purpose(Зорилго тодорхойлох)

Дээр дурдсан гурван алхамыг ашиглан өгүүлбэрийн зорилгыг олох боломжгүй. Жишээлбэл, 'Чи зуун метр сэлж чадах уу?' өгүүлбэр нь сонсогчийн чадварын талаар асууж байхад . 'Та надад тусалж чадах уу?' гэсэн өгүүлбэр нь зүгээр л эелдэг хүсэлт юм. Англи хэл дээрх өгүүлбэр нь мэдээлэх, хүсэлт тавих, амлах, лавлах гэх мэт олон янзын зорилготой байж болно. Тэдгээр зоригуудын аль нь болохыг тодорхойлохын тулд прагматик дүн шинжилгээ хийх шаардлагатай

Removing ambiguity(Илүү тодорхой болгох)

Зарим хэд хэдэн утга илэрхийлдэг үгнүүд бий. Мөн ижил дуудагддаг хоёр өөр үг ч бий шүү дээ. Тэдгээр нь syntactic, semantic шатуудаар ялгагдаж чадахгүй. Тэгвэл прагматик дүн шинжилгээ хийх өөр нэг зорилго нь тэдгээр тодорхойгүй байдлыг арилгах юм.

Хайх



Зураг 12. Хайлтын орон зайн жишээ

SEARCHING

Хиймэл оюун ухааны асуудлыг шийдвэрлэх аргуудын нэг бол хайлт бөгөөд хайлтыг олон тооны төлөв (нөхцөл байдал) ашиглан асуудлыг шийдвэрлэх гэж тодорхойлж болно. Хайлтын процедур нь эхний байрлалаас эхэлж, зорилтот байрлалд хүрэх хүртэл завсрын төлвүүдээр дамждаг. Жишээ нь, оньсого тайлахад эхний төлөв нь тайлагдаагүй оньсого, завсрын төлөвүүд нь оньсого тайлахын тулд хийсэн алхамууд, зорилтот төлөв нь оньсого тайлагдсан нөхцөл байдал юм. Хайлтын процесст ашигладаг бүх төлөвийн багцыг хайлтын орон зай(search space) гэж нэрлэдэг. Зураг 12-д таван төлөвтэй хайлтын орон зайн жишээг үзүүлэв. Аль ч муж нь эхний эсвэл зорилтот орон зай байж болно.

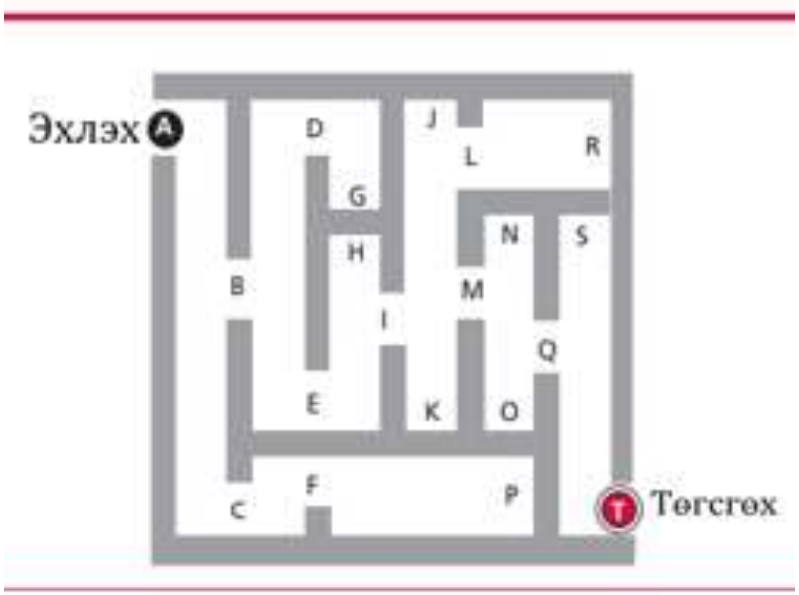
Хайх : Bruto force search

Search methods

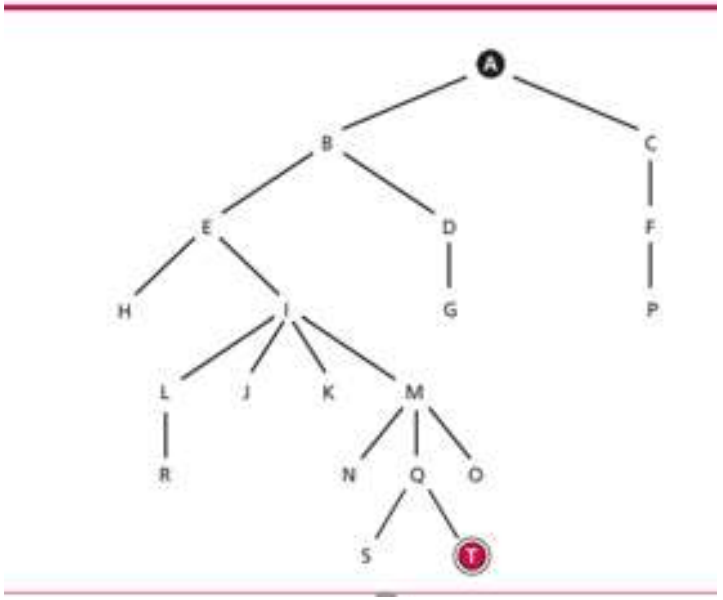
Хайлтын ерөнхий хоёр арга байдаг. : *brute-force* болон *heuristic*. breadth-first болон depth-first гэсэн 2 хайлт нь brute-force төрлийн хайлт юм.

Bruto-force search

Хайлтын талаар урьдчилж мэдэх зүйл байхгүй бол энэ аргыг ашиглана. Жишээлбэл:



Зураг 13a. brute-force хайлтын арга



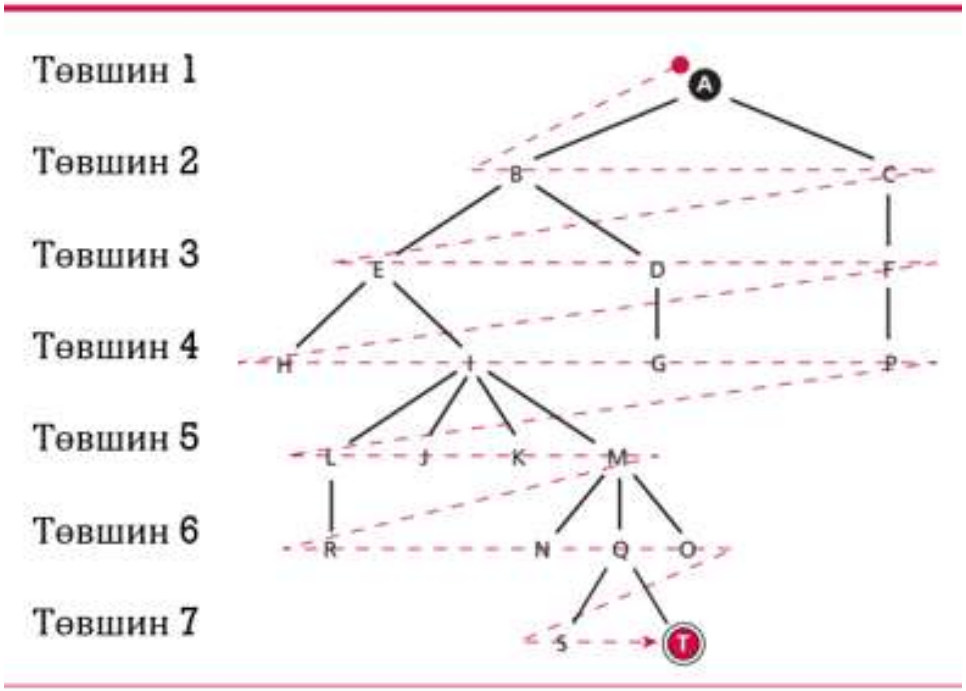
Зураг 13b. Мод бүтэц нь

Зураг 13. дээрх төөрдөг байшинг мод болгон харуулав

Хайх : Breadth first search

Breadth-first search

Энэ аргын хувьд бид модны үндэснээс эхэлж, дараагийн түвшинд шилжихээсээ өмнө түвшин бүрийн бүх зангилааг шалгадаг. Төөрдөг байшинг зүүнээс баруун тийш хайх Breadth-first search-ыг Зураг 14-д үзүүлэв.

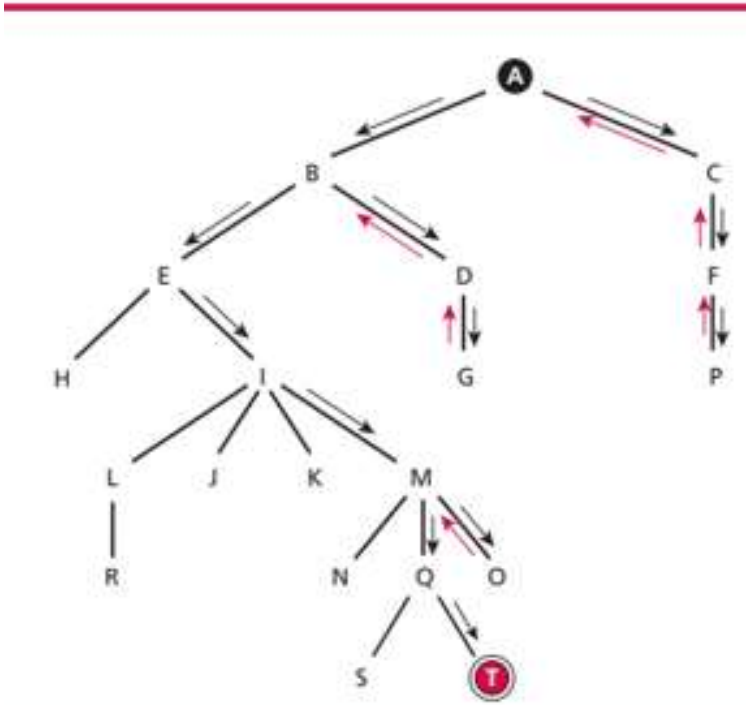


Зураг 14. Breath-first хайлт

Хайх : Depth first search

Depth-first search

Энэ аргад бид модны үндэснээс эхэлж, зорилгодоо хүрэх эсвэл мухардалд орох хүртэл явсаар байна. Хэрэв бид мухардалд орвол хамгийн ойрын мөчир руу буцаж очоод дахин хайлт хийнэ. Зорилгодоо хүртэл энэ үйл явц үргэлжилнэ.



Зураг 15. Depth-first хайлт

Хайх : heuristic search

Heuristic search

Heuristic хайлтыг ашиглан бид node бүрт *heuristic утга* (h утга) гэж нэрлэгддэг тоон утгыг онооно. Энэ тоон утга нь node-ын зорилтот төлөвтэй харьцангуй ойртлыг харуулна. Жишээлбэл:

Зураг 16. Эхний ба зорилгын төлөвүүд

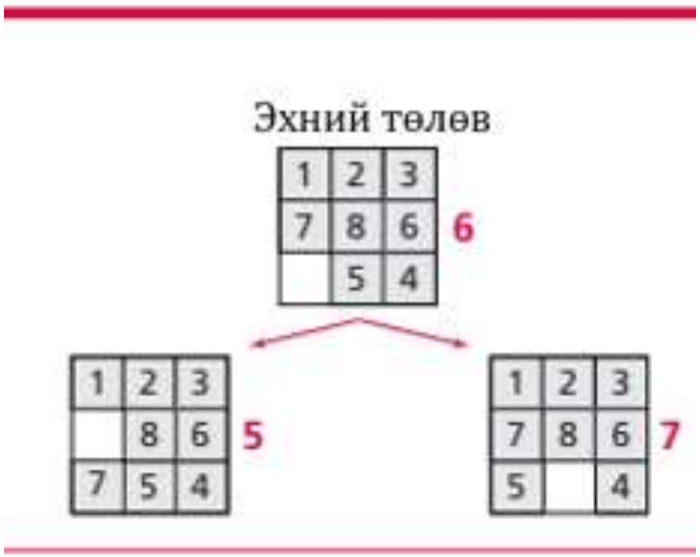
Оньсогоны анхны болон зорилтот төлөвийг харуулсан байна гэж бодъё. Нүд тус бүрийн heuristic утга нь зорилтот төлөвт хүрэхийн тулд нүдний хийх ёстой үйлдлийн тоо(хамгийн бага) юм. Төлөв бүрийн heuristic утга нь тухайн төлөв дэх нүднүүдийн heuristic утгуудын нийлбэртэй тэнцүү. Хүснэгт 16-т оньсогоны эхний болон эцсийн төлөвийн heuristic утгыг харуулав.

Хайх : hearsitic үнэлгээ

Зорилгын төлөвийн боломжит төлөвийн утга

Тайлын дугаар	1	2	3	4	5	6	7	8	Нийт
Эхний төлөвийн боломжит төлөвийн утга	0	0	0	1	1	2	1	1	6
Зорилгын төлөвийн боломжит төлөвийн утга	0	0	0	0	0	0	0	0	0

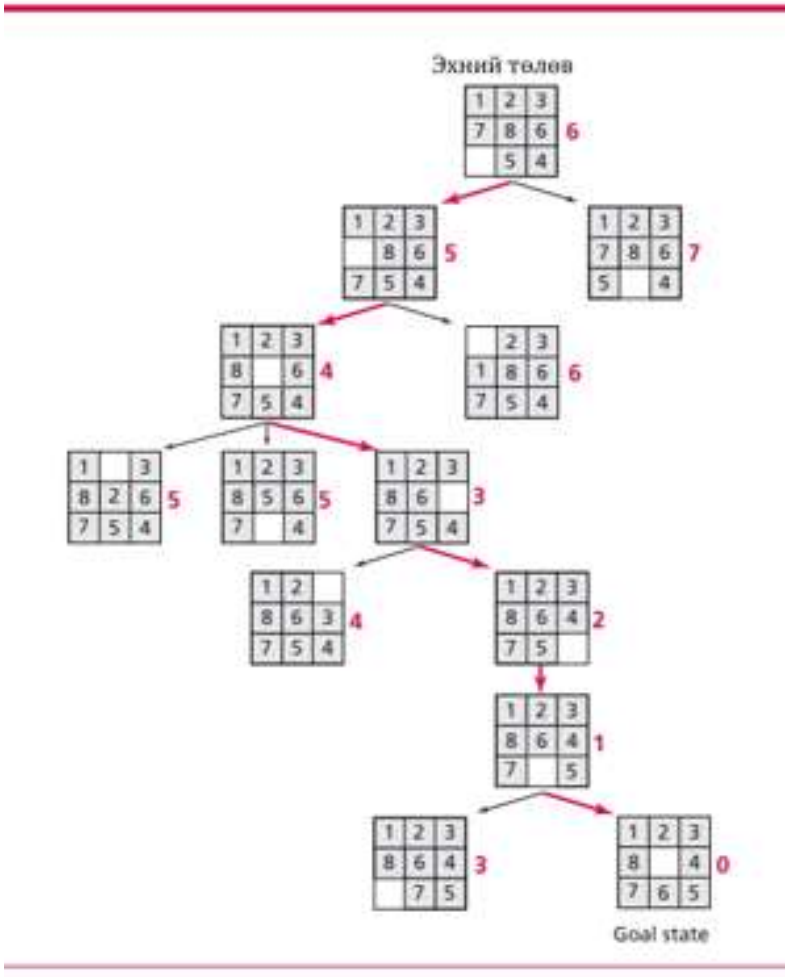
Хайлтыг эхлүүлэхийн тулд бид дараагийн түвшний бүх боломжит төлөвүүд болон тэдгээрийн харгалзах heuristic утгыг авч үздэг. Бидний оньсогоны хувьд нэг нүүдэл нь зөвхөн хоёр боломжит төлөвийг үүсгэх бөгөөд h утгыг Зураг 17-д үзүүлэв.



Зураг 17. Эхний төлөв

Хайх : төлөв

Дараа нь бид h утга багатай төлөвөөс эхэлж, дараагийн түвшний боломжит төлөвүүд рүү шилжинэ. Бид 18-р зурагт үзүүлсэн шиг h утга тэг (зорилтот төлөв) төлөвт хүрэх хүртлээ энэ замаар явна.



Зураг 18. 8-puzzle -ийн хайлт

NEURAL NETWORKS

NEURAL NETWORKS

Хиймэл оюун суралцах гэдэг зүйлийг ойлгож байж л сая л нэг хүн шиг авирлаж эхлэх байх. Гэтэл сурах нь хүмүүс бид нар өөрсдөө хүртэл бүрэн ойлгодоггүй биологийн нарийн төвөгтэй үзэгдэл билээ. Тиймдээ ч хиймэл оюуныг сургах ажиллагаанд оруулна гэдэг нь тийм амар ажил биш гэдэг нь ойлгомжтой. Гэсэн хэдий ч өнгөрсөн хугацаанд ирээдүйд итгэл найдвар төрүүлэхээр хэд хэдэн аргыг ашиглаж байжээ. Тэдний ихэнхи нь *inductive learning* эсвэл *learning by example* аргуудыг ашигласан байдаг. Энэ нь маш их хэмжээний асуудлыг шийдэлтэй нь хамт машинд оруулна гэсэн үг. Энэ хэсэгт бид эдгээр аргуудын зөвхөн нэгийг нь авч үзэх болно: neural networks. Neural networks нь мэдрэлийн эсийн сүлжээг ашиглан хүний тархины сурах үйл явцыг дуурайхыг оролддог.

NEURAL NETWORKS : биологийн эс

Biological neurons

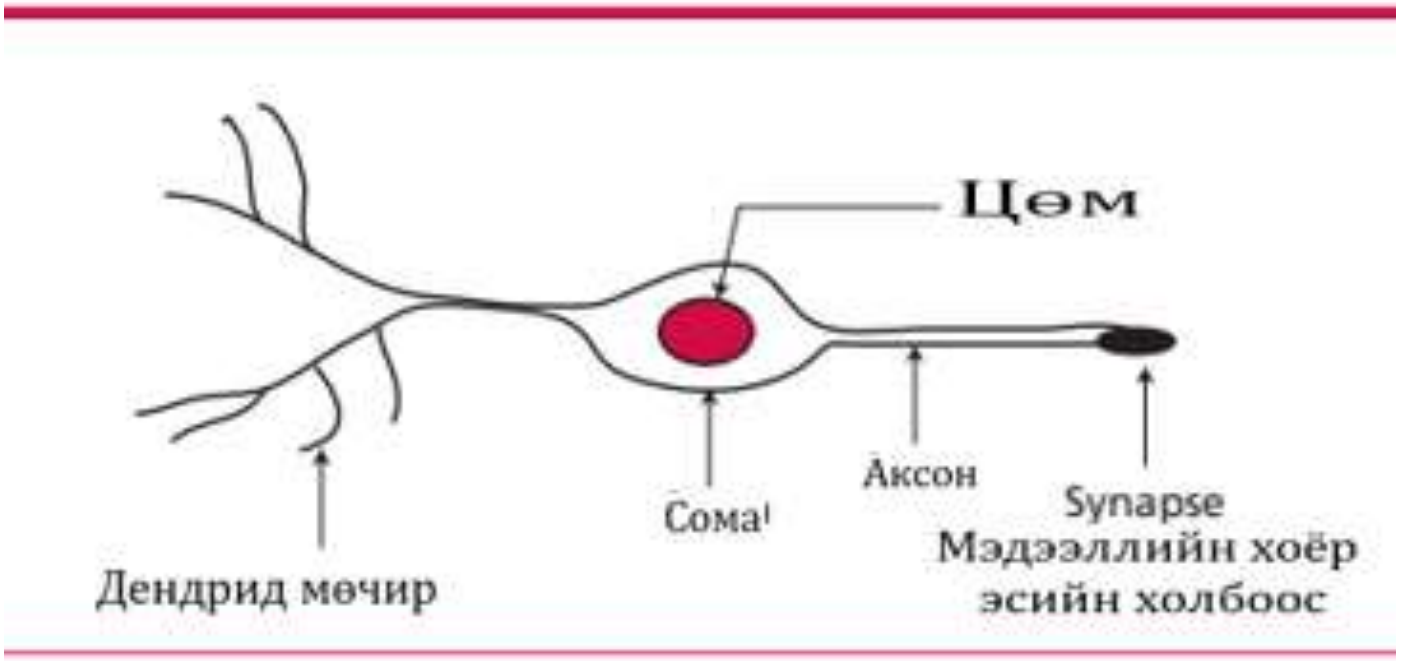
Сома (бие) нь эсийн цөмийг агуулдаг: энэ нь процессор юм. Дендрит нь оролтын төхөөрөмжийн үүрэг гүйцэтгэдэг: дендрит бүр өөр мэдрэлийн эсээс оролт хүлээн авдаг. Аксон нь гаралтын төхөөрөмжийн үүрэг гүйцэтгэдэг: гаралтыг бусад мэдрэлийн эсүүд рүү илгээдэг. **Синапс** нь мэдрэлийн эсийн аксон ба бусад мэдрэлийн эсийн дендритүүдийн хоорондох холболтын цэг юм. Дендритүүд нь хөрш мэдрэлийн эсүүдээс цахилгаан дохиог цуглуулж, сома руу дамжуулдаг. Синапсын үүрэг бол хөрш нейрон руу дамжих дохионд жин тавих явдал юм: энэ нь үүссэн химийн материалын хэмжээнээс хамааран хүчтэй эсвэл сул холболтын үүрэг гүйцэтгэдэг.

Нейрон нь excited(өдөөгдсөн) эсвэл inhibited(саатсан) гэсэн хоёр төлөвийн аль нэгэнд байж болно. Хэрэв хүлээн авсан дохио нь тодорхой хэмжээнд хүрвэл сома өдөөгдөж, аксон руу дамждаг гаралтын дохиог ажиллуулж, эцэст нь бусад мэдрэлийн эсүүд рүү дамждаг. Хэрэв хүлээн авсан дохио нь хангалттай хэмжээнд хүрэхгүй бол нейрон нь саатсан төлөвт үлддэг: энэ нь гаралт үүсгэхгүй.

NEURAL NETWORKS : биологийн эс

Biological neurons

Хүний тархинд нейрон гэж нэрлэгддэг хэдэн тэрбум боловсруулах үүрэг бүхий эсүүд байдаг. Нейрон бүр дунджаар хэдэн мянган мэдрэлийн эсүүдтэй холбогддог. Нейрон нь 19-р зурагт үзүүлсэн шиг сома, аксон, дендрит гэсэн гурван хэсгээс бүрдэнэ.



Зураг 19. Нейроны энгийн бүтэц

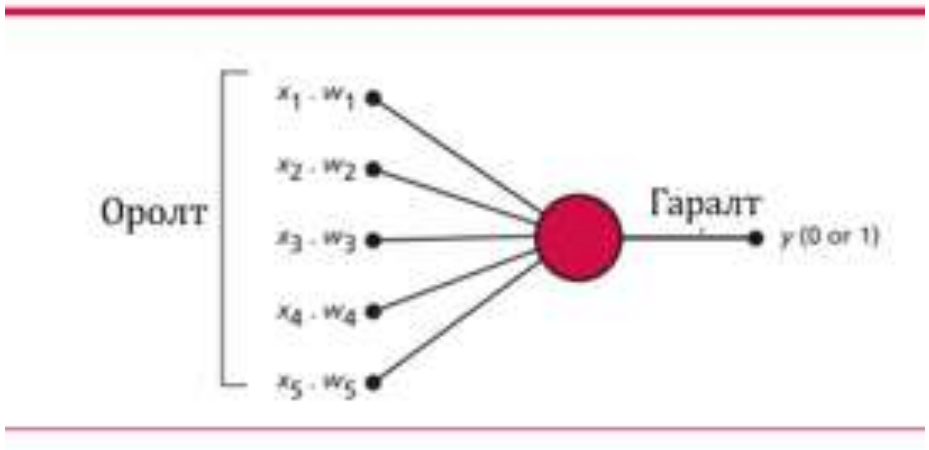
NEURAL NETWORKS : Мэдрэхүй

Perceptrons

Перцептрон нь нэг биологийн нейронтой төстэй хиймэл мэдрэлийн эс юм. Энэ нь оролтыг хэмжиж нэгтгэн, үр дүнг threshold-тай(босго утга) харьцуулна. Хэрэв үр дүн нь threshold-оос давсан бол перцептрон асдаг, үгүй бол асахгүй. Перцептрон асах үед гаралт нь 1: асаагүй үед гаралт нь тэг болно. Зураг 20-т таван оролт (x1-ээс x5), таван жинтэй (w1-ээс w5) бүхий перцептроныг үзүүлэв. Энэ перцептрон дээр хэрэв T нь threshold бол гаралтын утгыг дараах байдлаар тодорхойлно.

$$S = (x_1 \cdot w_1 + x_2 \cdot w_2 + x_3 \cdot w_3 + x_4 \cdot w_3 + x_5 \cdot w_5)$$

If $S > T$, then $y = 1$; else $y = 0$

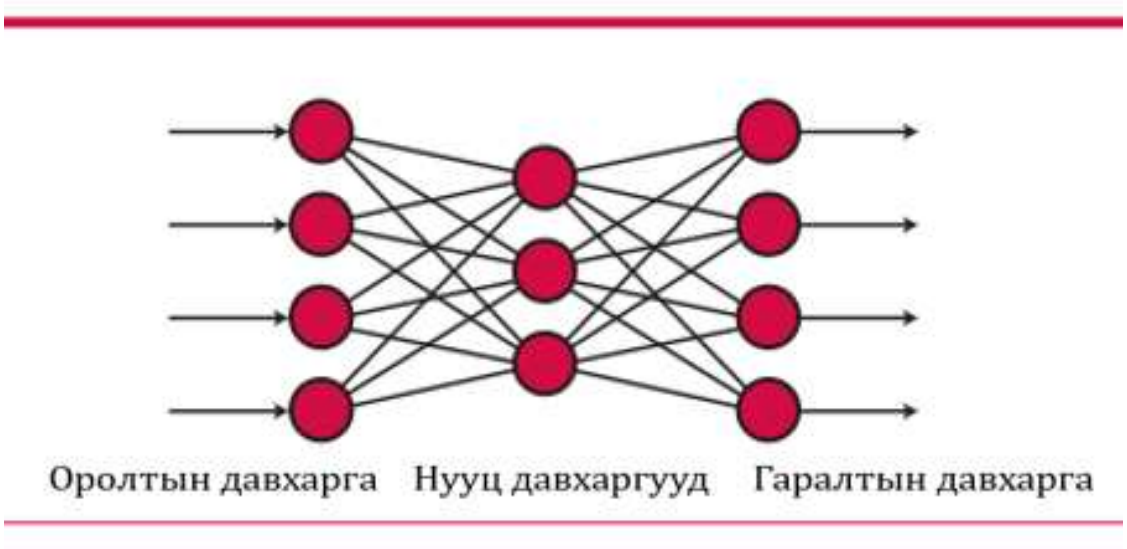


Зураг 20. A perceptron Персептрон

NEURAL NETWORKS. Олон түвшинт сүлжээ

Multilayer networks

Олон давхаргат мэдрэлийн сүлжээг бий болгохын тулд ойлголтын хэд хэдэн давхаргыг нэгтгэж болно. Давхарга бүрийн гаралт нь дараагийн давхаргын оролт болно. Эхний давхаргыг оролтын давхарга, дунд давхаргыг далд давхарга, сүүлчийн давхаргыг гаралтын давхарга гэж нэрлэдэг. Оролтын давхарга дахь node-үүд нь нейрон биш, зөвхөн тараагчид юм. Далд node нь өмнөх давхаргын гаралтыг хэмжихэд ашиглагддаг. Гурван давхарга бүхий мэдрэлийн сүлжээний жишээг Зураг 21-т үзүүлэв.



Зураг 21. Олон түвшинт нейрон сүлжээ

АШИГЛАСАН МАТЕРИАЛ

Foundations of Computer Science, Behrouz A. Forouzan, Fourth Edition, Cengage Learning EMEA, 2018

Бүлгийн гарчиг Artificial Intelligence

АНХААРАЛ ХАНДУУЛСАНД БАЯРЛАЛАА