

人工知能A

Topic 9: 知識表現 (Knowledge representation)

9

知識表現

- 知識表現の手法について概説する。
- 講義の内容
 - プロダクションシステム
 - 意味ネットワーク (知識グラフ)
 - フレーム (参考のみ)
- この他にも、述語論理（推論と結びつけて）もあるが、ここでは講義のスキープのスキープで割愛する

2

3

知識表現の概要

知識の活用

- 人間は統計的学習のように、非常に多数の例を見なくても学習や認知をして、間違いを減らしたり、判断を早めることができる。多少であれば、推測もできる。
- 非常に高度な知識（専門知識など）を獲得できる。
- 推論や学習の理由や根拠を説明できる。
 - 人間の知性を計算機で実現したいという人工知能の目的からすると、このような推論、推測（予測）、学習を解明し、アルゴリズムとして実現する必要がある。
- 人間は学習し、言葉（記号）として記憶している（ものが多い）。
 - 知識を記号として計算機内に表現し、それを利用した推論や学習を実現する。
 - その他「時間の概念」、「因果関係」、「他人に関する知識」なども統計的な処理だけではできない。

4

知識表現の必要性

- 知識の記述 — 知識表現 (knowledge representation)
 - 人間は、知識を用いて、外部からの情報や状況判断をしながら知的活動を進める。
 - 復習：これがエージェントの定義でもありました。
 - 知的活動を起こさせるには、計算機に知識を何らかの形で表現する必要がある。
 - 計算機システムに実現された知識の集まりを **知識ベース (knowledge base)** と呼ぶ。
 - 多くの場合、言語的に（記号で）記述可能なものを知識ベースというが、広くは記号では表現しない（e.g. ニューラルネットワーク）ものなど他の表現方法も含めることもある。
 - 当初名前はデータベースに対応したものであるが、データベース自体も高度化し、広い意味では大きな区別はなくなりつつある。

5

知識表現の必要性

- 主に記号ベース
 - ⇔ 統計的、ニューラルネットワーク
 - 何故必要か？
 - エージェント間や人との通信で、明示的な内容が必要。
 - エージェントが人間や他のエージェントと組む社会性を実現するには必須となる。
 - ただし両者の組合せは重要
 - たとえばニューラルネットワークの学習では、明らかに禁止されていると分かる行為の学習にも、多数の教師データが必要であり、それでも間違えることもある。
 - たとえば禁止事項をルールとして加えて、禁じ手を回避（除外）することも考えられる。

6

知識表現の条件

- 知識表現に求められる条件と特徴
 - 未整理、しかも膨大な知識を扱わなくてはならない
 - たとえば電車に乗るための全知識といわれてもなかなか表現できない
 - 切符を買う、ホームに移動するなど、場面ごとに切り出せば知識は表現できるかもしれない。
 - 知識の追加、修正、削除が容易である必要も？
 - 推論の成否により変更することがある。
 - 規則やデータにあいまいさがある
 - 鳥は飛ぶ（正確には、ほとんどの鳥は飛ぶ）。
 - 風邪をひくと熱がでる（こともある）
 - 計算機上のデータ構造 (モデル) であることは必須

7

記憶のモデル (1)

- 人の記憶モデル（二重貯蔵モデル, dual storage model, Atkinson & Shiffrin 1968）を土台に
 - 短期記憶(short-term memory, 動作記憶=working memoryとも言う)
 - 目や耳などの五感からの情報の一時的貯蔵という考えと、対話や学習行動などの認知活動に必要な情報の処理のための短期的記憶との立場
 - 長期記憶(long-term memory)
 - 宣言的記憶(declarative memory)と手続的記憶(procedural memory)に分類
 - 宣言的記憶
 - 言語的な表現が可能なもの。
 - 手続的記憶（非宣言的記憶）
 - 動作や活動の認知として表現される知識（自転車に乗るなど）
 - AIの分野では、記憶されるものを広く知識として扱い、宣言的知識、手続的知識などという。

8

記憶のモデル (2)

- 宣言的記憶の例
 - エピソード記憶 (episodic memory、特定の場面や時間、出来事など経験に結びついた記憶)。ある場面でになると記憶の引き出しから関連する情報を引き出すというイメージ
 - 意味記憶 (semantic memory、一般的な概念的知識)。「酸素は O_2 」とか、「犬はほ乳類」とかの一般的な知識。
- 手続的記憶 (非宣言的記憶)
 - 手続き (たとえば料理の手順、(不)定積分の求め方など) として表現される知識。
- プライミング
 - ある刺激がそのあとの処理を促進する。感覚が影響を受けたり、関連知識が出現しやすくなるような、連想記憶的な要素。

9

宣言的知識、手続的知識の例

- 例：
 - 車の運転席にある機器の構成：ハンドル、アクセル、スピードメータ、などは長期記憶の宣言的知識
 - 車の運転の仕方：「ハンドルを操作してまっすぐ走る」は長期記憶にある手続的知識
 - 「スピードメータを見たら、いま55km/hで走っている」、「ガソリンは満タンだ」などは短期記憶

10

知識表現の種類

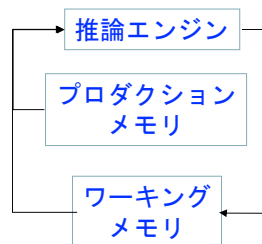
- プロダクションルール
 - IF-THEN形式の表現。あいまいさを導入できる。二重貯蔵モデル。
- 意味ネットワーク (その後、知識グラフ、wordnetなど)
 - 知識をネットワーク (グラフ) として表現。意味記憶。
- フレーム
 - 場面ごとに知識をまとめる (エピソード記憶)
- 述語論理、非単調論理、時相・様相理論
 - 数学的な述語論理式で知識を表現する。推論規則がある。
 - 非単調論理：知識やデータが追加されると結論が変わる (これを非単調性という) 論理表現
- ニューラルネットワーク (知識表現の一方法とも)
 - ネットワークの重みとして学習結果を記憶

11

12 プロダクションシステム

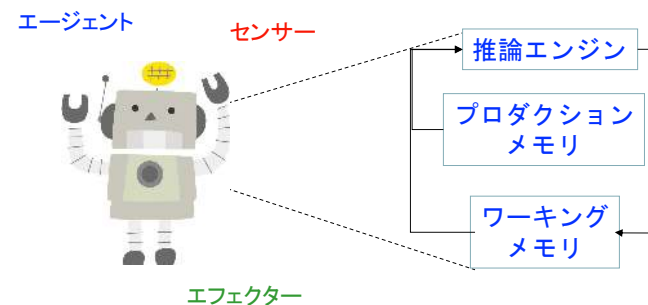
短期記憶と長期記憶

- プロダクションシステム
 - Newellによって1973年に提案。人間の問題解決行動に関する研究をもとに、計算機用を実現されたもの。
- プロダクションシステムの構成
 - 長期記憶に相当する知識ベース（もしくはプロダクションメモリ、PM）。規則や一般的データの集合
 - 短期記憶に相当する作業記憶（ワーキングメモリ、WM）。観測事象や推論の（途中）結果などを一時保存。
 - WMの情報にPMの知識を適用して新しい結果を導く推論エンジン（推論機構）



13

プロダクションシステムを内在するエージェント



- 初期状態、ゴール、センサーのデータ等をワーキングメモリに蓄え、プロダクションメモリにあるルール（知識）を使って推論を行い、行を選択する

14

プロダクションシステムの構造 (1)

- プロダクションメモリ(PM) – ルールベースとも言う。
 - 下記のIF-THEN形式のルールの集合として構成される

$$C_1, C_2, \dots, C_m \rightarrow A_1, A_2, \dots, A_n$$
 - C側を条件部またはLHS (left hand side)、A側を実行部またはRHS (right hand side)という。これは、短期記憶を参照し、「LHSが成立するなら、RHSを実行せよ（これは行為であったり、短期記憶を書き換えたりする）」と解釈する。
 - このルールを**プロダクションルール (production rule)**と呼ぶ。動作としてはsendmail.cfのルールみたいなもの。
- 例：ライトが暗い、セルモータがよく動かない、バッテリーは新しい
 → 発電機が故障している
- 曖昧さを表現するためにルールに確信度がつく場合もある。
- 例：ライトが暗い、セルモータがよく動かない、バッテリーは新しい
 → 発電機が故障している [0.7]

15

プロダクションシステムの構造 (2)

- ワーキングメモリ(WM)
 - 初期状態、外部世界からの観測したデータ、推論の途中データ、到達したいゴールなどの**事実**が表現される。
 - 例：ライトが暗い
 セルモータがよく動かない
 バッテリーは古いなど。
 ゴールは「車を修理したい」など
- 推論エンジン
 - プロダクションルールとワーキングメモリの内容から、新たな結果を導く。通常は、ゴールに向かって（ゴールを達成するために）推論を行う。
 - **前向き推論**（ゴールに向かう）と**後ろ向き推論**（ゴールからそれを達成するためにというルールの繋がりを探る）。

16

後ろ向き推論 (backward reasoning)

- 後ろ向き推論の動作
 - ゴールをWMに与える。
 - 実行部を参照しゴールを直接達成できるルールをPMから選択する。次にその条件部とWMを参照し、WMに含まれない条件をとりだす。
 - その各条件を達成することを実行部に含むルールを選択する。
 - さらに、これらのルールの条件部とWMを参照し、WMに含まれない条件を取り出し、それを達成するルールを探す.....を繰り返す。
 - 上記の繰り返しの最後は、条件部が他のルールの実行部により達成できないときもある。そのときには、環境から情報をセンサーなど取得するか人間に直接問い合わせをして、WMに格納する。
 - ゴールを与えることで推論が開始されるので、**ゴール駆動推論 (goal-driven reasoning)**ともいう。
 - また、後ろ向き推論の場合は、実行部で実際に実行を伴わないので、通常は実行部を**結論部**と呼ぶ。

17

プロダクションルールの例

- P1: 体毛を持つならば、哺乳動物である
P2: 授乳するならば、哺乳動物である
P3: 飛ぶことができ、産卵するならば、鳥である
P4: 羽をもち、ペンギンでないならば、飛ぶ
P5: 哺乳動物であり、肉食ならば、肉食動物である
P6: 哺乳動物であり、鋭い歯を持ち、鋭い爪をもつならば、肉食動物である
P7: 哺乳動物であり、蹄を持つならば、有蹄動物である
P8: 肉食動物であり、黄褐色で、黒い縞を持つならば、トラである
P9: 肉食動物であり、黄褐色で、黒い斑点を持つならば、チーターである
P10: 有蹄動物であり、黄褐色で、黒い斑点をもつならば、キリンである

内部的には、アンダーラインの部分は変数として表せる。たとえば、

P1: \$体毛=yes → \$哺乳動物=yes (\$で変数を表す)

P6: \$哺乳動物=yes, \$歯=鋭い, \$爪=鋭い → \$肉食動物=yes

18

後ろ向き推論の例 (1)

- 体毛を持ち、鋭い牙と爪を持ち、黄褐色で黒い縞がある動物はトラであるか？
 1. 上記のアンダーラインの部分は事実としてWMに追加される。ゴールはトラであることを示すこと。
 2. 結論部にトラがあるのはP8であるので、その条件部を示せばよい。
 3. WMを参照し、黄褐色で黒い縞を持つことは分かっている。残りの肉食動物であることを示す。
 4. 結論部に肉食動物であることを含むのは、P5とP6である。
 5. 同様にして、P5条件部とWMを参照すると、ともにWMには入っていない。以下同じくP1を使って哺乳動物であることは分かるが、肉食であるかは分からない。
 6. そこでP5の代わりにP6を考える。鋭い歯と爪を持つことはWMにあり、また哺乳動物であることは、P1を使えばわかる。よって、上記は証明された。

19

後ろ向き推論の例 (2)

- 前の例の項番5で、肉食かどうか分からない場合に、人間に尋ねるという方法もある。
 - この場合、「その動物は肉食ですか？」という問いになる。
- 人は、「はい」「いいえ」「わからない」の答えがあり、
 - 「はい」ならP5が成立。
 - 「いいえ」ならP5は成立しない。
 - 「分からない」なら一応P5は成立とするが、他のルールを試みる。他のルールで証明できなければ、条件つきでトラであることと、その条件（肉食であるなら）を提示して終了する。

20

後ろ向き推論の例 (3)

- ある動物が目の前にいて、それが何かを知りたいとする。この場合は、トラかチータかキリンであるかを順に示す。
- 動物に関する情報はWMにないので、必要になれば人に問い合わせ結果を入力してもらう。
 - たとえば初めにキリンを示そうとして、「蹄を持ちますか？」と問合わせる（結果はWMに入る）。これを繰り返しキリンと証明できなければ、次にチータ、それに失敗すればトラであることを順に証明する。
- 問い合わせ中心のシステムは、動物当てでは不要かもしれないが、故障診断（たとえば自動車）では有益である。
 - 車のすべての状況を初めに入力できないので、「ライトは暗いですか？」「エンジンはかかりますか？」などと調べる項目を順に示すことができる。

21

後ろ向き推論の特徴

- 後ろ向き推論は、ゴールを成立させるようにWMを参照しながらPMのルールを逆向きにたどる。WMにない事実は、人に問い合わせたり、環境から観測することもある。
- 特徴と問題点：
 - ルールを逆向きにたどるだけなので、効率がよく、実装も容易
 - PMを適用してWMを書き換えることはない（問い合わせでは書き換わる）
 - 基本的に環境に対する行為は問い合わせか観測だけで、環境に影響を与えるような行為を一つ一つ行うような制御は不可能（たとえばロボットの行動など）。これは行為と逆方法に推論を行うため。ただし、推論の後にまとめて行為を行うことはできる。
 - 診断など、環境に影響を与えないものに使われる。たとえば、血液感染症の診断システムMYCINはこの推論方法を使った。

22

MYCINのルールの例

- MYCINは、血液感染症を診断し、有効な抗生物質を推奨するシステム（Stanford 大学、1970年代）

IF :

感染症が原発性菌血症であり、
培養検体採取部位が通常無菌と考えられる部位であり、
問題とする細菌が進入したと考えられる感染経路が消化器官である

THEN:

問題とする細菌の種類がバクテロイデスである。この確信度は0.7である。

- 性能が悪かったわけではないが使われなかった。人は機械が間違えると許せない。また、間違ったときに誰が責任を取るかという法律面の壁に当たった。

23

前向き推論 (forward reasoning)

前向き推論の動作

- 照合 (matching)
 - WMを参照しながら条件部が成立するルールをPMから探す。このようなルールの集合を競合集合といいCSと表す。
- 競合解消 (conflict resolution)
 - CSの要素がなければ終了。CSの要素が複数ある場合には、何らかの戦略(競合解消戦略)でひとつだけ選ぶ。この戦略は予め決めておく。
- 実行 (action)
 - 選択されたルールの実行部を実行する。WMが書換る
- プロダクションサイクル (Production cycle, PC)
 - 上記を各1回行うことをプロダクションサイクルという。実行によりWMが書き換わり、次のサイクルで別のルールが選択される。
- ゴールの状態になるまでPCを繰り返す。

24

前向き推論の例

- 先の動物当ての例題：「体毛があり、鋭い歯と爪を持ち、黄褐色で黒い斑点を持つ動物は何か？」
- $WM = \{\text{体毛、鋭い歯、鋭い爪、黄褐色、黒い斑点}\}$

PC1: $CS = \{P1\}$ であり、これを実行すると、
 $WM = WM \cup \{\text{哺乳動物}\}$

PC2: $CS = \{P1, P6\}$ ここでうまい戦略によりP6を選択
 $WM = WM \cup \{\text{肉食動物}\}$

PC3: $CS = \{P1, P6, P9\}$ ここでうまい戦略によりP9を選択
 $WM = WM \cup \{\text{チータ}\}$

となり、この動物はチータとなる。

25

前向き推論の注意事項

- 注意：
 - プロダクションサイクルに「実行」とあるが、これはWMの内容を変えるだけで、必ずしも環境への行為を意味するわけではない。あくまで、メモリにあるモデル（つまりWM）を書き換えるだけ。実際に、何時、実世界での行為を行うかは独立の問題である。
 - たとえば、ロボットであるなら、PCで右足を出すとしてWMを書き換えても、実際に右足を出すのは後でかまわない。数ステップ先まで考えて（WMを書き換えて）、うまく行くと判断してから実際の行為をとってもよい。
 - 後ろ向き推論と違い、推論を適当な場所で中断し、実際の行為を発行することは可能
 - 「うまい戦略」が不適切なルールを選択しても、バックトラック（推論を取り消す）はない。よって競合解消戦略の選択は重要。

26

前向き推論のポイント1 ー 競合解消戦略ー

- CSから適切なルールを選択すること（競合解消戦略）が重要となる。適切なルールの適用順序が結論を早く導く。
- 競合解消戦略の例：
 - ルールに重要度を予め与え、重要度の大きいものを優先する。
 - ルールの条件部の多いもの（より詳細と考えられる）を優先する。
 - 最も最近使われていないものを優先する。
 - 同じルールは続けて使用しない。
 - 最近WMに加えられた事実を使用する条件を持つものを優先する。
 - 実行して、WMに新しい事実が加わるものを優先する。
- 通常は、上記の戦略を組み合わせる。
- 課題9-1：先の動物当ての例と同じようにCSからルールを選択する戦略（の組み合わせ）を2、3考えよ。

27

前向き推論のポイント1 ー 競合解消戦略ー

- 有名な推論戦略の例ー LEX戦略 (Lexicographic sort)
 - (1) すでに実行したルールをCSから除去する。
 - (2) WMの最新のデータ（事実）を参照するルールをCSから選択する。
 - (3) ルールの条件部の個数が多いものをCSから選択する。
 - (4) 任意のルールをCSから選択する。
- (1) から順に戦略を適用し、ルールが一つになった時点で処理を終了する。もし、(1)の適用後CSが空集合になれば、推論は終了する。
- これはあくまで一例であり、戦略は問題領域やプロダクションルールの作り方に依存する。

28

前向き推論のポイント2 – 照合

- WMを参照して、条件にあうルールをPMから効率的に選択する必要がある（PCごとに照合が行われる）。
- リートアルゴリズム (Rete algorithm)
 - ルールを適用しても、実行部によってWMは少ししか変わらないと仮定する。
 - PMをコンパイルしてリートと呼ばれるネットワークを生成する（Reteは（体内の）網という意味。神経網、動脈網などの網）
 - ルールを適用するたびにWMが変わり、それにあわせてリートを書き換えるが、その変更はわずか。網を参照すると条件部を満足するルールが簡単に分かる。
 - 本方式を使ったプロダクションシステムとしてOPS5があり、OPS5を利用したシステムにR1（のちにXconと改名された）がある。初の商用化エキスパートシステム。

29

参考：OPS5 and R1/XCON

- OPS5 (Official Production System 5)
 - Invented by Charles Forgy, カーネギーメロン大学 (CMU)
 - Reteアルゴリズムに改良を加えられ高速化したRete IIが提案（詳細は非公開）。これを実装したOPSJ (Java version)が売られている（現在も）。その後Rete IIIが大規模化に対応した。
- R1/XCOM --- DEC社（IBMと並び中心的存在だった）
 - VAXという巨大なミニコンピュータの構成を提案し、発注を助言
 - VAXは当時の中心的存在。VAX MIPSという現在のMIPS以外の概念もあった。
 - 実は、DECはR1の前に同様のシステムを一般的なプログラミング言語（BASICやFORTRAN）で作ろうとしたが複雑すぎて成功しなかった。OPS5を使ってこれを実現した。
 - 2500ルール。8万以上の適用事例があった。

30

リートアルゴリズムの例

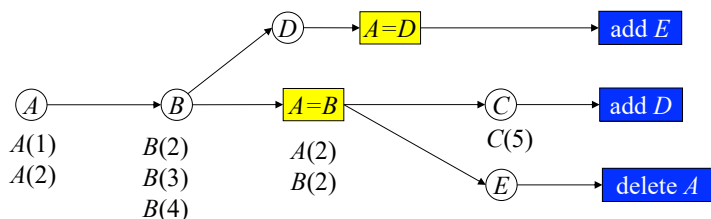
PM: $R_1: A(x) \wedge B(x) \wedge C(y) \rightarrow \text{add } D(x)$

$R_2: A(x) \wedge B(y) \wedge D(x) \rightarrow \text{add } E(x)$

$R_3: A(x) \wedge B(x) \wedge E(y) \rightarrow \text{delete } A(x)$

WM: $\{A(1), A(2), B(2), B(3), B(4), C(5)\}$

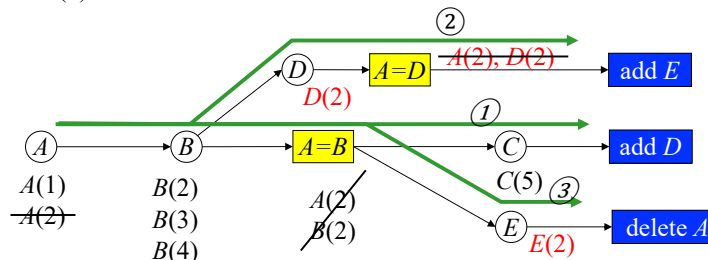
これから多く共通に含まれる条件（この場合は $A(x)$ ）から順にリートネットワークを作る（コンパイルする）。



31

ルールの適用とリートの書き換え例

- add Dに達するルート①がある（ R_1 に相当）ので、 R_1 を適用。
- add Dにより $D(2)$ が新規に加わる。
- add Eに達するルート②ができ、これに相当する R_2 を適用。
- add Eにより $E(2)$ が新規に加わる。
- delete Aに達するルート③ができ、これに相当する R_3 を適用。
- $A(2)$ が消える（この場合は、ネットワークの変更が多くなる）。



32

リートアルゴリズムの性質と問題点

- 各ルールの実行部でWMの内容が変わるが、それは一度の変更量は少ないことを想定している。多数の実行部を持つルールが多数あると効率はいさ少し下がる。
- 多くのルールの条件部にある事実に変更があると、修正がネットワーク全体に広がり、修正に時間がかかる。
- 必要メモリ量は大きめ。
- しかしWMを毎回参照してルールを選択するよりは効率的で、現在でも有効な手段として使われている。
 - ルール形式の表現は分かりやすく、ルールベースの管理システムなどで使われている。
 - 参考文献：Charles Forgy, "Rete: A Fast Algorithm for the Many Pattern/Many Object Pattern Match Problem", *Artificial Intelligence*, Vol. 19, pp 17-37, 1982.

33

説明機能

- 推論に利用したルールを提示して、結論の根拠を説明。
 - 例：「体毛があり、鋭い歯と爪を持ち、黄褐色で黒い斑点を持つ動物は何か？」
 - この結論はP9を使って得られた。どうしてチータかと理由をたずすと、P9を提示し、システムは

「肉食動物であり、黄褐色で、黒い斑点を持つならば、チータである」というルールがその根拠である

と提示でき、さらにどうして肉食動物と分かるのかという問いには、P6を提示し

「哺乳動物であり、鋭い歯を持ち、鋭い爪をもつならば、肉食動物である」というルールがその根拠である

と示せる。

- 複雑な問題解決の根拠を示すという意味で、説明機能は人工知能アルゴリズムとしては重要な機能である。

34

プロダクションシステムのまとめ

- 長所
 - 意味的にIf-Then形式のルールは分かりやすい。
 - ルールは独立しており、ルールの追加、変更が独立に行うことができる。
 - 前向き推論と後向き推論の両方が可能である。
 - 説明機能があり、ルールを提示することでその根拠を示せる。
- 短所
 - 独立性の反面として、知識の相互関係が分かりにくい。特に推論システムは、決められた戦略に基づいて自動的にルールを選択するので、関連のある知識、同時に考慮したい知識などと関係なく実行される可能性がある。
 - 系統だった知識を記述できない（基本的に平坦的な記述となり、後に示すような階層的な表現は含まれていない）。

35

課題

- 課題 9-1
 - 記憶の記憶の二重貯蔵モデルに基づいて考えられた知識表現方法は？
 - 二重貯蔵モデルとは、_____記憶と_____記憶からなる。
 - プロダクションシステムは推論エンジンと_____と_____からなる。
 - プロダクションシステムの推論方式には_____推論と_____推論がある。
 - 前向き推論における照合、競合解消について説明せよ。
 - 前向き推論を高速に実行するアルゴリズム名を書け。
 - プロダクションシステムにおける説明機能の方法について述べよ。

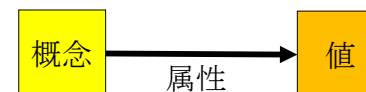
36

37 意味ネットワーク

Semantic Network (現在の Knowledge graph, Linked dataの始まり)

意味ネットワーク

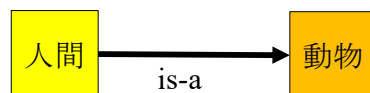
- 意味記憶、特に連想記憶という心理学的モデルをベースにQuillianによって1966年ごろに提案。
- 連想記憶とは：ある概念や対象（オブジェクト (object) ともいう）を考えると、それが持つ性質や属性(attribute)の内容や値(value)を通して、他の概念や対象が連想される。
- これを形式的に〈概念、属性、値〉（形式的には $\langle o, a, v \rangle$ ）として下記のような有向グラフで表す。



- 話題：ただしこれ以前(1901年)、一階述語論理の別の表記法として、上記と同じ記述法を持った存在グラフというものも提案されていた。
cf. Ross Quillian, *Semantic Memory*, PhD dissertation, Carnegie Mellon University, 1966.

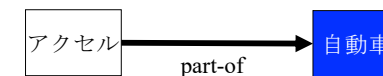
意味ネットワーク

- 意味ネットワーク（semantic network, 日本語ではセマンティックネットワークとも言う）
- 前のスライドの図により、概念の属性や概念間の関係を表し、全体としてネットワーク構造となる。
 - 節点（ノード）⇔ 概念、対象（物）、状況、属性値
 - 枝（エッジ、リンク）⇔ 属性、関係（最近はメタデータとも言う）
- 例：



重要な概念間の関係

- エッジに対応する属性や関係は、知識ベースのデザインの裁量であるが、いくつかの重要で不可欠な属性がある。
- 概念的階層構造
 - is-a (またはa-kind-of, ako) 関係と言われ、概念間の上位一下位の関係を表す。前ページの例では、人間という概念は動物という概念に含まれることを意味する。
 - この関係に基づいて継承（後出）という値の伝播が行われる。
- 構造的階層構造
 - part-of (またはhas-as) とも言われ、全体一部分の関係を表す。全体と部品の関係でもある。
- オブジェクト指向言語（C++やJavaなど）では、上記の構造が混在して使われるが、知識表現では明確に分ける。



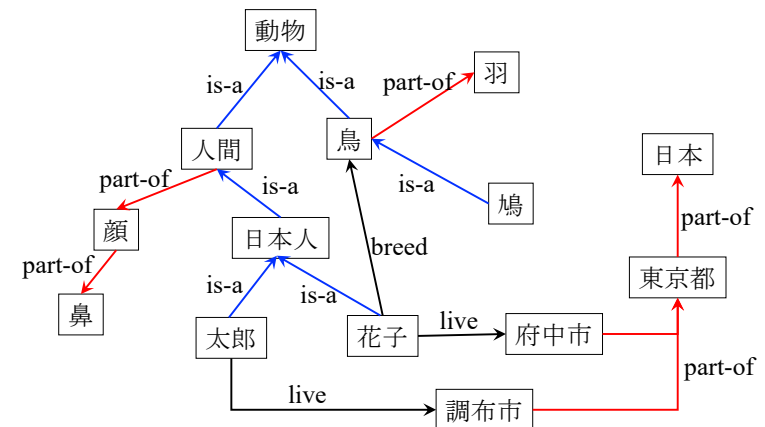
意味ネットワークの例

■ Quillianの例に基づく意味ネットワーク

- 1) 人間は動物である。 ⇔ is-a 関係
- 2) 人間には顔がある。 ⇔ part-of 関係
- 3) 顔には鼻がある。 ⇔ part-of 関係
- 4) 太郎と花子は日本人である。 ⇔ is-a 関係
(日本人は人間である) ⇔ is-a 関係
- 5) 花子は鳥を飼っている。 ⇔ breed 関係
- 6) 鳥は動物である。 ⇔ is-a 関係
- 7) 鳥には羽がある。 ⇔ part-of 関係
- 8) 鳩は鳥である。 ⇔ is-a 関係
- 9) 太郎は調布市に住んでいる。 ⇔ live (live-in) 関係
- 10) 花子は府中市に住んでいる。 ⇔ live (live-in) 関係

41

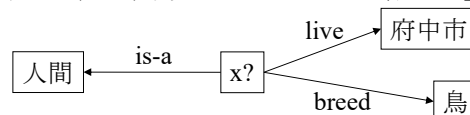
意味ネットワークの例



42

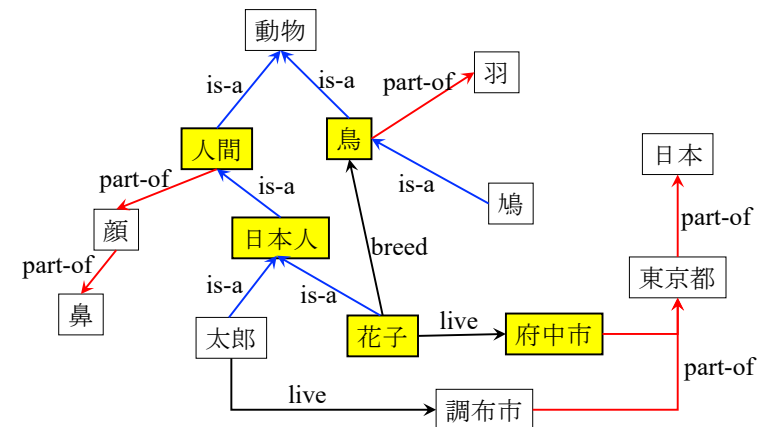
意味ネットワークを使った推論 (1)

- 意味ネットワークは知識を表現するだけで、それを使って推論する方法までは規定されていない（これは自由であり、面倒なところでもある）。
- 通常は、ネットワーク（グラフ）の照合を行う。
 - ここでグラフの照合は、属性の意味に合わせてまとめることもある。たとえば、is-aのis-aはis-aが成立（推移律）。part-ofも同様。そのほかの属性に関しては、そのデザインの意味から推移律が成立するかを考える。
 - 定義した属性や関係に関する属性の推移律も考慮。
 - 例：「府中市に住み、鳥を飼っている人は誰か？」



43

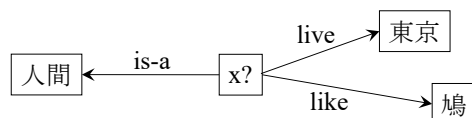
意味ネットワークの検索例 (1)



44

意味ネットワークを使った推論 (2)

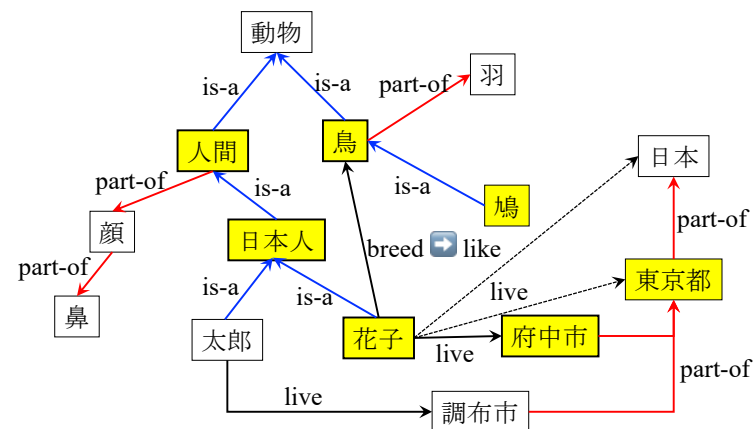
- 例：「東京に住み、鳩が好きな人は誰か？」



- Liveという属性は、part-ofを伝える。(府中に住んでいるということは、東京に住んでいる) これはシステムデザイナーが定義したliveの性質として与える。
- Breedという属性があると、自動的にlikeという属性も成立する(飼っているなら好きであろう)。これもシステムデザイナーが定義したliveの性質として与える。

45

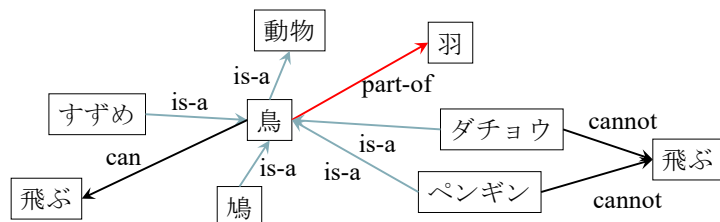
意味ネットワークの検索例 (2)



46

継承 (inheritance)

- 継承：is-a関係を通して、他の属性と関係は、否定されない限り伝達されること。



- canの否定がcannotであることは予め与えておく。
 - 「動物である」「羽を持つ」はis-a関係を通じて全ての鳥に継承される
 - が、「飛ぶ」に関してはダチョウとペンギンで否定されているのでこれらは飛べないが、それ以外は継承により飛べる。

47

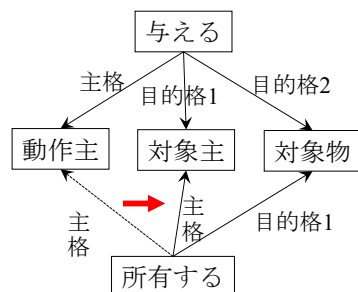
継承の解釈

- 例外を含む値と解釈する
 - 鳥の例：通常は飛べるので、鳥の一種であれば飛べるとしてもほとんど正しい。ペンギン、ダチョウなどの例外だけ付け加えればよい。
 - 無脊椎動物：無脊椎動物は硬い殻で守られていることが多い(昆虫、貝など)。しかし蛸などの頭足類は殻を持たない。しかし、頭足類でオウム貝は殻を持つ。これは、無脊椎動物のノードに殻を持つことを定義し、頭足類のノードにこれを否定する殻を持たないを定義する。さらにオウム貝のノードには、それを否定して殻を持つことを追加すれば、例外を段階的に定義できる。
- 継承された値は暗黙値と解釈できる
 - 暗黙値(default value)とは、特に指示されていないときに使われる値である。ある属性や性質が不明なときに、代用として使う。
 - 知識ベースでは、一般には正しいと考えられる性質や値を暗黙値として定義し、例外のみ記述することで、記述量を減らせる。

48

動作の記述

- 静的な記述だけでなく、動作を記述したい。
- 動作の主格、目的格を属性関係で結び動作を表す。
 - 例：「与える」は、「所有する」の主格を変える。

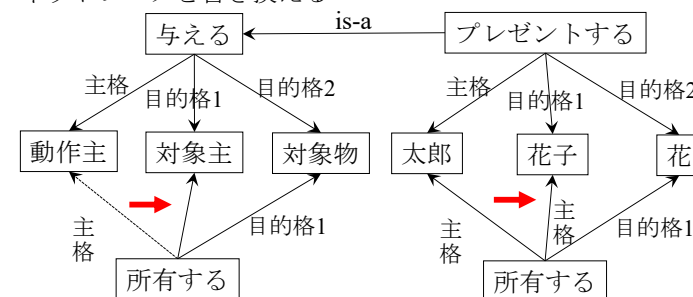


- 与えると所有するの関係を推論規則として表現したもの。
- 前提条件はグラフの照合により、行為はグラフの張替規則として表現する。

49

動作（ネットワークの書き換え）

- 太郎が花子に花をプレゼントする。動的に下記のネットワークを構成する。
- プレゼントするは与えるの特別な場合である。これをis-aで結ぶ。
- 「所有する」の部分が成立するかネットワークを参照して確かめる。
- ネットワークを書き換える



50

意味ネットワークのまとめ

- 概念 (対象)、属性 (関係)、属性値の関係を有向グラフで表す。
- 長所：
 - ネットワーク構造で、図としては見やすい。
 - 知識の追加は容易である（ネットワークを付け加えるだけ）
 - 矢印（ポインター）をたどれるので、推論や処理は速くできる。
- 短所：
 - 推論規則はあまり強力ではない。別途用意することが多い。
 - 知識の矛盾は発生する（ただし、矛盾の記述が必要なこともある）。
 - 知識の一貫性などの検証はできないか難しい。
 - 知識が大きき大規模なシステムでは、複雑さが増す。

51

52 意味ネットワークの派生表現方法

どちらかといえば、意味ネットワークの発展系

活性化拡散モデル

- ある概念に着目すると、それに近い概念が着目（活性化）されるような仕組みを導入。
- 基本的には意味ネットワークと同じだが、各エッジに距離（認知距離）のようなものを付与する。
- 着目している概念に近いものの方が活性度が高い。活性度で着目する順番が決まる。

53

知識グラフ (knowledge graph)

- 意味ネットワークのエッジの属性の種類を制限し、代わりにグラフ理論の形式的な操作が可能としたもの。
- 意味ネットワークの一種で有り、例で見たとおり推論などでグラフ操作は必要なので、現状では同じものと考えることが多い。
- 類似概念：Googleのナレッジグラフ
 - 検索性（というか知識グラフの一例）
 - セマンティック（意味）を考慮した検索を実現するために、関係性（リンクなど）に意味（属性）を付与。
 - 検索キーワードに関連する項目を関係でまとめる（この後に述べるフレームにも近いかもしれない）。
- 詳細は情報検索やWWWの講義で。
 - ここでは概要だけ少し。


54

インターネット上の知識グラフ

- 知識グラフに関する関連概念
 - URL: Universal resource locator
 - (みなさんがよく知っているURL。リンクにより情報（知識）の在処を関連づけている)
 - URI: Universal resource identifier
 - URLの拡張概念。情報+計算資源（ネットワーク上の機器やサービスなど）の場所の他、単純に概念（名前や識別情報など）。例：都市名、過去の人名などを差し示す。URLを含む概念。
- RDF: Resource Description Framework
 - URIの関係（メタデータ）を記述する枠組み。まさにセマンティックネットワークを表現するもの。
 - `<subject relation object>`の形式で、*subject*と*object*はURI。*relation*も実はURIで、どのような関係かを記述している（記述している場所を指し示す）。*relation*は、*predicate* (述語)ともいう

55

インターネット上の知識グラフ

- 当初、ネットワークのノードは、計算機上のテキストであり、それで多様な知識を表現することは難しかった。
- 現在、ノードは主にWWW上の知識とリンクで表現され、またそれを目指してURIやRDFの概念ができた。これを利用した例としてGoogleのknowledge graphがある。
- 先出しになるが、次のフレームは、このようなネットワーク形式の知識を概念ごとにまとめようという考えに基づく。
- たとえば、*subject*を一つ固定して、整理された形式になる)

```
<subject relation1 object1>
<subject relation2 object2>
<subject relation3 object3>
<subject relation4 object4>
<subject relation5 object5>
```



```
<subject relation1 object1,
relation2 object2,
relation3 object3,
relation4 object4,
relation5 object5>
```

56

57 フレーム

フレーム (Frame)

- 1974年M. Minskyによって提唱。エピソード記憶と関連し、ある場面や情景を思い浮かべると、それと関連した時間、空間、事柄などが記憶から連想されること。
- 整理された知識をフレームという一般的な枠組みで表現し、そこに関連する知識、関連する概念との関係、その事象に結びついた手続き（動作、処理）などの情報をまとめる。
- たとえば、自分の部屋を思い浮かべる。
 - その部屋に関するさまざまなものが関連して表現される。
 - たとえば、「窓」といえば一般的な窓ではなく、その部屋にある特定の窓。「ぬいぐるみ」と言えば、ぬいぐるみ全般ではなく、その部屋にあるクマのぬいぐるみ、というように自然と特定のものが結び付けられる。

58

フレーム (Frame)

- フレームの作成・編集などの環境を提供するシステムをフレームシステムという。
 - だいたい LISP (list processor) という言語で書かれていた。
 - LISPについてはあとで少し説明予定。
- 1980-90年代は多くのフレームシステムが販売されていたが、現在は見当たらない。
 - 基本的にはオブジェクトとしてオブジェクト指向言語で記述可能。

59

フレームの構造と種類

- フレームはいくつかのスロットとスロット値、また各スロットには任意の個数のファセットとその値からなる。

```
<frame>
  <slot1>   <slot value1>
    <facet1-1> <facet-value1-1>
    .....
    <facet1-N> <facet-value1-N>
  <slot2>   <slot value2>
    <facet2-1> <facet-value2-1>
    .....
    <facet2-N> <facet-value2-N>
  .....
```

 - さしあたり、ファセット部分は省略する。
 - 基本的にはスロットとスロット値のペアの集まりがフレームである。
 - スロットは属性名、スロット値はその属性値を表している
- 一般的概念を表す**クラスフレーム**と、特定の対象を表す**インスタンスフレーム**がある。
 - 一般的な「部屋」ならクラス、あなたの住んでいる特定の部屋ならインスタンスである。

60

フレームの例（ファセット省略）

Bird ; the concept “bird” (frame)
 wing: Yes ; has wings (slot)
 lay-egg: Yes ; lay eggs (slot)
 legs: 2 ; has two legs (slot)
 flies: yes ; can fly (slot)
 size: ; unknown (slot)
 name: ; unknown (slot)

← 一般的な鳥という概念
 に対応するフレーム
 (クラスフレーム)

ある特定の鳥を表現
 するフレーム（イン
 スタンスフレーム）

Bird1 ; a certain bird
 wing: Yes ; has wings
 lay-egg: Yes ; lay eggs
 legs: 2 ; has two legs
 flies: yes ; can fly
 size: 70 ; 70mm in length (体長70mm)
 name: warbler ; 鳥の名前

61

IS-AとMember (Instance-of) 関係

- フレーム間をIS-Aとmember関係で結ぶ。
 - スロットIS-A, Memberなどを作り、そこにスロット値として指し示すフレーム名を格納する。
- $A \rightarrow B$ とIS-A関係で結ばれている場合はA、Bともにクラスフレームで、概念Bは概念Aに含まれることをしめす。
 - 動物 \rightarrow 鳥 \rightarrow カナリアなど。AKO (a kind of) 関係ともいう。
- $A \rightarrow B$ とmember関係で結ばれている場合は、Aはクラスフレーム、Bはインスタンスフレームで、インスタンスBは概念Aの特定の対象物であることを示す。
 - カナリア \rightarrow warblerなど
 - 犬 \rightarrow ぼち など
 - Instance-of関係とも言われる。

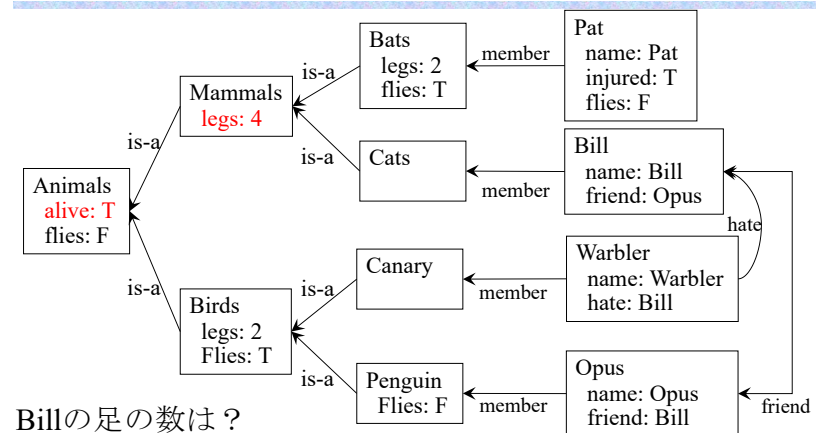
62

IS-A・Member 関係と継承

- IS-AとMember関係を通じて、スロットの値が継承される。いくつかの継承の方法が設定されている。たとえば、
 - 特に定義がなければ、上位概念の値がそのまま継承される（データが無い場合のデフォルト値とも考えられる）
 - 上位概念で与えられた値が集合（や数値の区間）となっており、その下位はその部分集合でなくてはならない。
 - 上位概念では集合が与えられているが、インスタンスフレームではそのうちの一つでなくてはならない。
- 1.は意味ネットワークと同じ考えである。2.3.は具体的な値を与えなければいけない
- 実際のフレームシステムでは、上記のような継承の仕様はインヘリタンスロール (inheritance role, 継承仕様) という値で区別する。特殊なファセットの値でこの違いを管理している。ファセットはあとで。

63

継承の例



Billの足の数は？
 Opusは生きているか？

64

多重継承 (multiple inheritance)

- IS-AやMember関係を複数可能とすると、多重継承の問題がある。
 - 多重継承にしたい例：戦闘機は飛行機であり武器でもある。iPhoneは電話かつ動画（音楽）再生器かつスケジューラである。
- 同名のスロットを持つ複数のフレームから継承された場合、継承方法に多様性がある。たとえば、
 - 和集合を取る場合 (union inheritance)
 - 共通集合を取る場合 (intersection inheritance)
 - 一方のみを選択する (selective inheritance)
- 一般には両方の可能性がある。また、内容が矛盾する場合もあり、どちらかを優先させることになる。これは応用依存となる。
- 複雑な取り決めになり、わかりにくくなる恐れもある。

65

手続き的知識

- これまでスロットの値は宣言的知識（つまり静的なデータ）のみを表現していた。
- スロット値に手続き的知識を格納できる。
 - 手続き的知識とは、プログラムであり関数でもある。付加手続き (attached procedure, procedural attachment) と呼ぶ場合もある。
 - このスロットにメッセージを送るとその手続きが起動する仕組み。（参考：オブジェクト指向言語）
 - フレームシステムの実装言語のは主にLISPであった。
 - LISP（関数型言語）では、プログラムがプログラムをデータとして生成し、それを定義・実行することができる。またプログラムをラムダ式で書けるので（というかLISPではプログラムはすべてラムダ式）、そのまま実行できる。
 - 参考：Scheme --- 計算理論の理解によい（近年ラムダ式は、Python, Ruby, C#, C++, 最近はJava8などの言語でも部分的に導入されている）。

66

ファセット (Facet)

- ファセットとは、スロットに関する属性や付随情報を記述する領域。その利用方法は、たとえば、
 - インヘリタンスロール：ファセット名をinheritance-roleなどとし、その値に応じて継承の仕様や制限をフレームシステムがチェックする。
 - データタイプ：スロットのデータの型チェック。ファセット名をdatatypeとして、たとえば整数 (integer) というファセット値を与えると、フレームシステムは、その値の属性のみがスロットに格納できるように制限する。
- デーモン (daemon, demon = 守り神, 次のスライドで。)
 - 上記2つの機能もこれで実現できる。
 - UNIX系のOSでデーモンとは、常駐してシステムを常時管理・制御しているソフトウェア群のことだが、それとは違うので混同しないように。
 - ただし「出番を待ち受けて必要なときに動作する」という広い意味では似ている。

67

デーモン (daemon)

- デーモン：特別な名前のファセットを定義し、その値に手続き的知識を格納すると、対応するスロット値に何らかのアクセス、変更が加わる前後にその手続きを起動できる。
- 利用法：
 - 値がないときに代用値(default値)を計算する。（これは継承によるもの以外。たとえば、人に関するフレームで体重が分からないとき、身長と性別などの情報から標準値を暗黙のうちに計算する）
 - 値に変更があったときに、それと関連した他のスロット値との間に矛盾が起これないように設定する。

68

手続き的付加とデーモン

- ファセットの例
 - If-added: スロット値が追加されたときに起動する。
 - If-needed: スロット値を求める（アクセスした）ときに起動する。
 - If-removed: スロット値が消去されたときに起動する。
 - 上記の3つはデーモンの例。
- 以下もある条件で起動する手続きであり、付加手続きの一例である。
 - default: 代用値を計算する。
 - require: 値が格納されるときに、値として正しいか（データの種類を規定）を判断する。たとえば、人のフレームで身長は正数であるなど。

69

フレームと意味ネットワーク

- 意味ネットワークの関係（ノード間のエッジ）は、主ノードをフレームのノードにエッジ名（関係名）のスロットを作り、その値にエッジの反対側のノードを記載する。
 - 意味ネットワークで定義したpart-ofは、”part-of”というスロットで、part-of関係にあるフレームをスロット値として現す。
 - この観点からすると意味ネットワークの表現はフレームでも可能である。ただし、フレームは継承（is-aとmember-of関係）に基づく構造化が明確に定義されている。その意味で、知識は整理されている。
- フレームシステムによっては多重継承も可能。ただしこの場合、何を継承するかを明確にする。
 - 複数の継承の一方だけ、継承の内容の和（積）集合、継承内容が条件であればand/orのどれかなど。

70

フレームのまとめ

- フレームシステムは典型的な知識をフレームという形式で関連するものをまとめて表現できる。
- 宣言的知識が中心だが、スロットに手続き（プログラム）を格納し、手続き的な知識に対応できる。
- 継承により性質の記述をまとめることができる。
- 汎用性はあるが、大規模な知識体系をフレームのみで表現するのは難しい。
- フレーム間の矛盾管理は困難。フレームシステムの利用者に任せられている。

71

課題

- 課題 9-2
 - 意味記憶（連想記憶）をベースに作られた知識表現方法は何か？
 - is-a関係と、part-of関係について述べよ。
 - 意味ネットワークのグラフ構造にグラフの形式的操作（演算）を適用することを想定したものを_____という。
 - 意味ネットワークにおける継承機能について簡単に述べよ。
 - URLについて述べよ。
- 課題 9-3
 - エピソード記憶のモデルに基づいて考えられた知識表現方法は何か？
 - フレームはいくつかの_____と_____値、また各スロットには任意の個数の_____とその値（_____値）からなる。
 - IS-A関係とMember関係について述べよ。

72

その他（論理的表現）

- その他、知識を表現するための多くの（かつ深い）研究がある。一部はデータベース理論などとも関係あり。
- たとえば、
 - **命題・述語論理**（論理表現だけではなく推論規則とも結びつく）
 - **様相論理 (modal logic)**：～の可能性がある（ない）などの、否定と肯定の中間的表現が可能。たとえば、「宇宙人はいる可能性がある」。（完全に肯定はできないが、否定もできていない）
 - **時相論理 (temporal logic)**：時間概念を含めた事実、たとえば、「あるときから以降ずっと成立する」、「あるときまで成立していた」などの表現を可能とする。様相論理の一種と考えられている
 - **線形時間論理**：時間の流れは一次元と仮定して論理を展開。Linear temporal logic
 - **計算木理論（分岐時間論理）**：未来は木のように分岐しているという考え。Computational tree logic

73

その他

- **認識論理 (epistemic logic)**：「他の人は何を知っているか？」「私がある事実Aを知っていることを友達は何を知っているか？」「友達はAを知っている可能性はあるか」など信念や共有知識を表現する。様相論理の一種と考えられている。
- 通信の信頼性 (coordinated attack problem, two general's problem)や、分散システムの理論、分散人工知能 (multi-agent systems)、分散データベースなどの理論的基盤としても使われている、
- **非単調論理 (non-monotonic logic)**：事実を追加すると推論結果も変わるかも知れない。
 - たとえば、「鳥は飛ぶ」が既知のときに、特に情報がなければ鳥の仲間のペンギンも飛ぶと推論する。ここに「ペンギンは飛べない」という事実を追加すると、ペンギンは飛べないと推論する（継承と類似したメカニズムを理論的に扱う）。フレーム問題に挑む。
 - **Default logic** (否定を帰結できない限りは信じるという表現を追加)
 - **Circumscription** (矛盾しない極小モデルの範囲を信じる)
 - **Autoepistemic logic** (自己認識論理、「事実Aを知らない」を表現する)

74