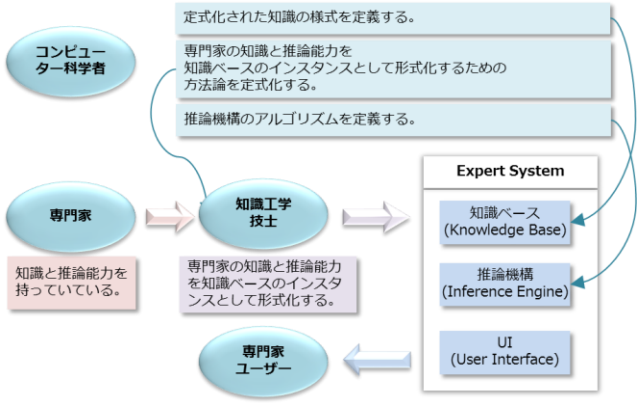
B-015

メタ知識の定義に基づく Expert System の一般化フレームワーク

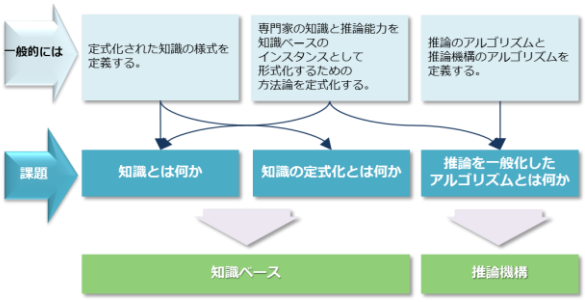
中村　正治
Masaharu NAKAMURA

1．まえがき

Expert System は、決定木のプログラム問題を解決するために、決定のアルゴリズムを「知識ベース」に収納し、アルゴリズムを駆動する「推論エンジン」を実装するシステムである、とされている。

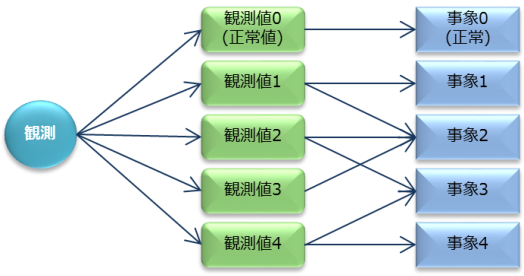


Expert System の一般的な構造を定義しようとすれば、その課題が前記 3 点に集約されることは明らかである。Expert System は、この課題に、知識ベースと推論機構という構成で応えようとした。

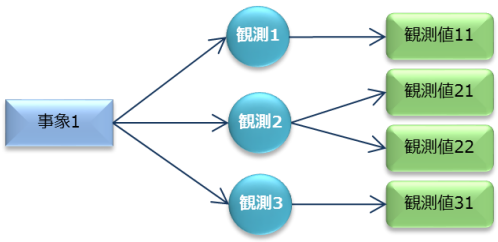


しかしながら、知識・推論という言葉自体の抽象度は極めて高く、議論を進めにくいので Expert System の一般性を少し落として、症状・診断システムと再定義することにする。

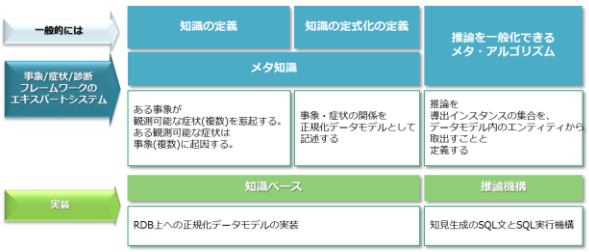
診断から原因となる事象をたどるのは後ろ向き推論で、図にすると下記のようになる。観測を中心にしたものは、後向き推論の図示と考えることがでる。



ある事象 が惹起している症状は、複数の Observable にわたるので、やはり図示でき、前向き推論の図示と考えることができる。



これらのことから、症状-診断として形式化した Expert System は、知識モデルとして正規化データモデルを使用すれば、特殊なプラットフォームや推論エンジンを必要とせず、エレガントに構築できる。



2．メタ知識

公理系　◧

定義　1　症状

◐：Observable 観測可能な対象の集合

◧：Eigenvalue 全ての Observable のとりうる値の集合

このとき、∀e∈ ◧ ∃O∈ ◐ : e↦ O とする。(全射)

e∈◧, O∈ ◐ のとき、E 1 ⊆ {e↦ O } なる部分集合を定義して、S=(E1, O)と書き Symptom 症状と呼ぶ。

定義　2　正常状態

∀O∈ ◐ について、S0=(E0, O) ∧ E0={e0} なる e0 を一つ定義し、正常な症状、ないし無症状と呼ぶ。

定義　3　事象と発症

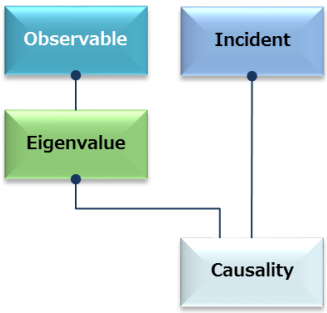
▌: Incident 事象

◐: Causality 発症

Incident, Causality　なる集合が存在し、

∀c∈◐ ∃ e∈ ◧ ∃ I∈ ▌で　e≠e0 のとき、c↦ (I, e) が存在する。

この公理系は、データモデルでグラフとして記述できる。



データモデル 𝔇 のグラフ

エンティティ・クラスが集合を、エンティティ・インスタンスが元を、子から親へのリレーションが子から親への全射を表している。

データモデリングの構成方法に従えば、

Observable, Eigenvalue, Incident が Primary であり、Cross Section として Causality が定義される。

メタ知識をデータモデル 𝔇 により定義すれば、推論は、データモデル 𝔇 上での集合操作として定義できる。すなわち、

前向き推論とは、**Incident** の ひとつのインスタンスが与えられた場合の、リレーションシップを満たす すべての **Eigenvalue** とその親の **Observable** のインスタンスの集合を構成することである。

後向き推論とは、**Observable** とその子の **Eigenvalue** のひとつのインスタンスが与えられた場合の、リレーションシップを満たす **Incident** のすべてのインスタンスの集合を構成することである。

データモデルは「表」として表現できる。したがって、上述の前向き推論・後向き推論は SQL により記述できることは明らかである。

Observable

id_Observable	Attributes・・・		
o1			
o2			
o3			
・・・			

Incident

id_Incident	Attributes・・・		
i1			
i2			
i3			
・・・			

Eigenvalue

id_Eigenvalue	id_Observable	Attributes・・・	
e1	o1		
e2	o1		
e3	o2		
・・・	・・・		

Causality

id_Causality	id_Eigenvalue	id_Incident	Attributes・・・	
t1	e1	i1		
t2	e2	i1		
t3	e3	i2		
・・・	・・・	・・・		

この公理系=データモデルは、理論上も実装上も有用であり、かつ何よりも、ミニマルな構造であることが評価できる。こ

さらに、現実に即して、この公理系を拡張することを考える。

公理系 𝔎

定義 1 症状

𝔒 : Observable 観測可能な対象の集合

𝔈 : Eigenvalue 全ての Observable のとりうる値の集合

このとき、∀e∈ 𝔈 ∃O∈ 𝔒 : e↦ O とする。(全単射)

e∈𝔈, O∈ 𝔒 のとき、 E 1 ⊆ {e↦ O } なる部分集合を定義して、 S=(E1, O)と書き Symptom 症状と呼ぶ。

定義 2 正常状態

∀O∈ 𝔒 について、 S0=(E0, O) ∧ E0={e0} なる e0 を一つ定義し、正常な症状、ないし無症状と呼ぶ。

定義 3 事象と発症

𝔍 : Incident 事象

𝔠: Causality 発症

Incident, Causality なる集合が存在し、

∀c∈𝔠 ∃ e∈ 𝔈 ∃ I∈ 𝔍 で e≠e0 のとき、c↦ (I, e)が存在する。

定義 4 症候群

𝔛 : **Indication** 兆候

𝔐 : **Syndrome** 症候群

∀x∈𝔛 ∃ M∈𝔐 ∃ e∈ 𝔈 で x↦ (e,M) なる組が存在するとき、M を症候群 Syndrome と呼ぶ。

このとき、 M=S({e : x↦ (e,M)})とかく。M, S({e : x↦ (e,M)})ともに、混乱がない場合は症候群と呼ぶ。

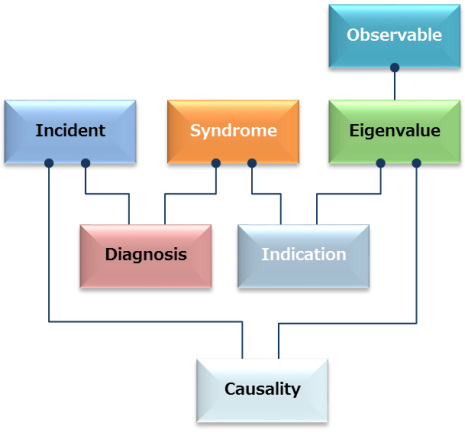
定義 5 診断

𝔊 : **Diagnosis** 事象

すべての症候群の集合を 𝔐 とする。

∀g∈𝔊 ∃ M∈𝔐 ∃ I∈ 𝔍 で g↦ (I,M) なる組がするとき、が存在するとき、gf=(I,M) と書き、診断 Diagnosis と呼ぶ。

𝔇 と同様 𝔎 もデータモデルにより、グラフで記述できる。



データモデル 𝔇 のグラフ

エンティティ・クラスが集合を、エンティティ・インスタンスが元を、子から親へのリレーションが子から親への全射を表している。

データモデリングの構成方法に従えば、データモデリングの構成方法に従えば、Syndrome, Eigenvalue の Cross Section として Indication が、Syndrome, Indication の Cross Section として Diagnosis が定義される。

公理系 𝔎 は公理系 𝔇 を包含している。Syndrome, Diagnosis, Indication が追加されて、より知識が「豊かに」なったといえる。同時に、原因からその惹起する現象を

推論するのが前向き、現象からその原因を推論するのが後向きという用語の本来の意味合いから、データモデルのエンティティが増えて複雑化した時、多様な推論とその実装である SQL が考えられるので、前向き、後向きという区分や定義に拘泥する必要はなくなる。

表による表現に関しては、データモデル D の表に、

Indication

id_Indication	id_Syndrome	id_Eigenvalue	Attributes	
x1	m1	e1		
x2	m1	e3		
x3	m2	e5		
・・・	・・・	・・・		

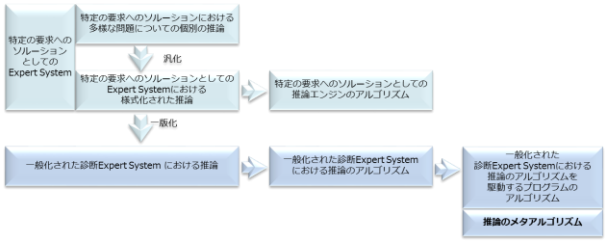
Diagnosis

id_Diagnosis	id_Syndrome	id_Incident	Attributes	
g1	s1	i1		
g2	s1	i2		
g2	s2	i3		
・・・	・・・	・・・		

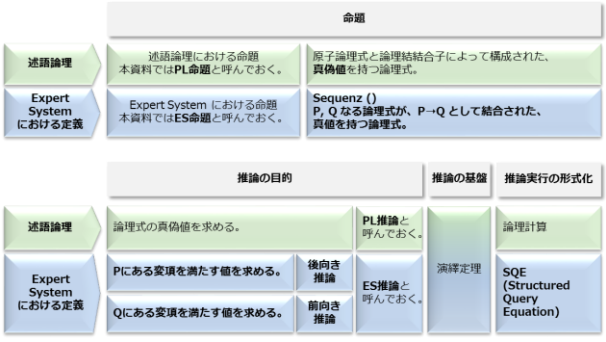
を付加したものとなっている。

3. メタアルゴリズム

メタ知識の定義は、特定の課題に対する専用の知識ベースを一般化したメタ知識によって記述し様式化する方法を提示した。そのメタ知識上で、専用のプログラミングに依存しない、メタ推論というべきものを定義しておく必要がある。

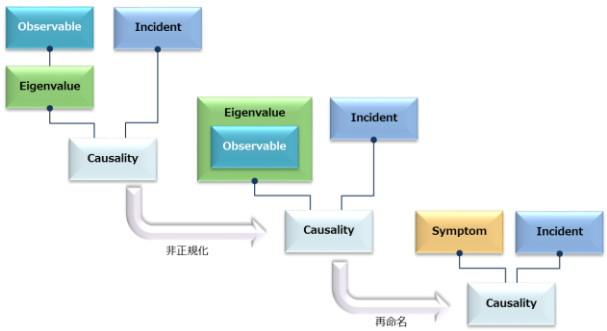


述語論理において、推論とは、真偽値を持った論理式から演繹定理によって、真偽値を持つ別の論理式を導くことである。述語論理に限らず命題論理においても、常に問題は真偽を求めることである。一方、Expert System における推論とは、Sequenz と呼ばれる形の真なる論理式が与えられた場合の変項 (あるいはその集合) 求めるものであり、論理学一般の推論という用語とはなじまない。



本論では用語の不用意な混用を避けるため、これ以降、必要に応じて、一般的な論理式の演繹操作を PL 推論、変項を求める操作を ES 推論と呼んで区別する。

公理系 D のモデルの取扱いを簡易にするために、Observable を Eigenvalue に繰込む、親の子への繰込みの非正規化を行った形の上で議論を進める。このンティティの呼称を Symptom と、改めて公理系 Dx と定義する。



公理系 Dx は、原因 Incident と結果 Symptom の因果関係を記述しているが、モデルは、双方に対象であり、推論の向きは、意味論的なものであって構造的なものではない。すなわち、P を Incident の命題、Q を Symptom の命題としたとき、Sequenz は P → Q であり、P の変項を求めるのが後向き推論、Q の変項を求めるのが前向き推論であるが、データモデルのグラフを見ると明らかなように、様式上は Incident と Symptom は対象であり、Sequenz の向きは、意味論的なものであり、ES 推論の構造を論ずる上では本質的なものではないことがわかる。

改めて、便利のため、数理論理学の記述により、集合として取り扱うことにして、

Incident の集合を I, Symptom の集合を S, Causality の集合を C とする。

P : C→I を親子関係の全射、Q : C→S を同じく親子関係の全射とする。

前向き推論

或る i∈ I について、

Cp={c∈ C | c↠ i }としたとき、{∀s∈ S | Cp∋ ∀c↠ s}

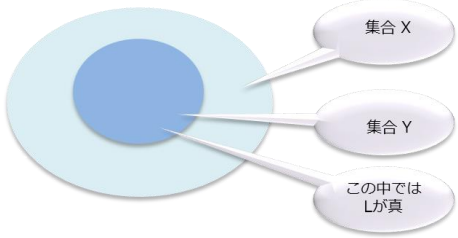
後向き推論

或る s∈ S について、

Cq = {c∈ C | c↠ s } としたとき、{∀i∈ I | Cq∋ ∀c ↠ i}

SQE

さてここで、SQE (Structured Query Equation) と称する、ES 推論の問題設定を記述する様式を定義する。



ある集合を X、真偽値を持つ論理式を L とするとき、

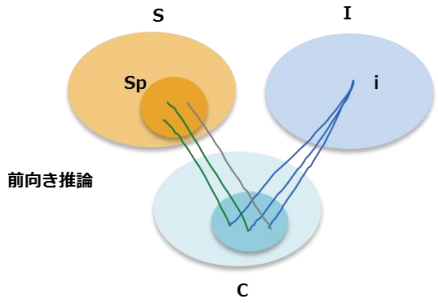
{∀x∈ X | L=True } なる X の部分集合 Y を、

Y=Select x from X where (L)

と定義し、この定義書式を SQE と称する。

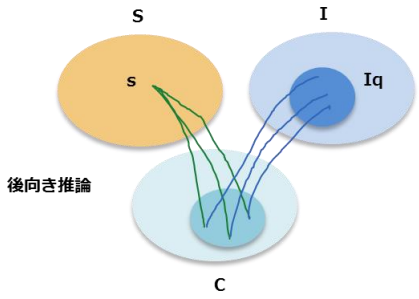
前向き推論は

{i}=Select i from I ⇒ Sp=Select s from S where (s∈ Q(c∈ Select c from C where (P(c)=i)))

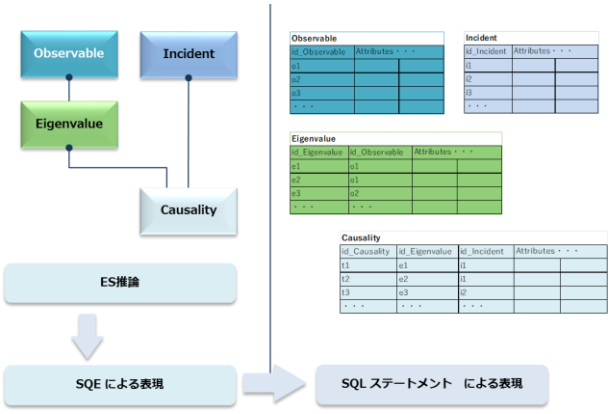


後向き推論は

{s}=Select s from S ⇒ Iq=Select i from I where (i∈ P(c∈ Select c from C where (Q(c)=s)))

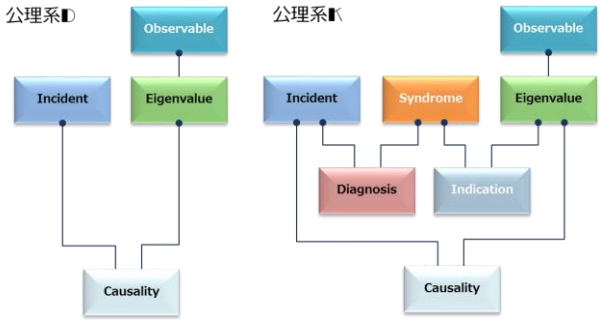


推論は、データモデル ◧ 上での集合操作として定義できる。データモデルの表現の一つに「表」があり、表の上での EQ 推論は、したがって、SQE を表の表現に合わせれば、SQL で表現できるがわる。

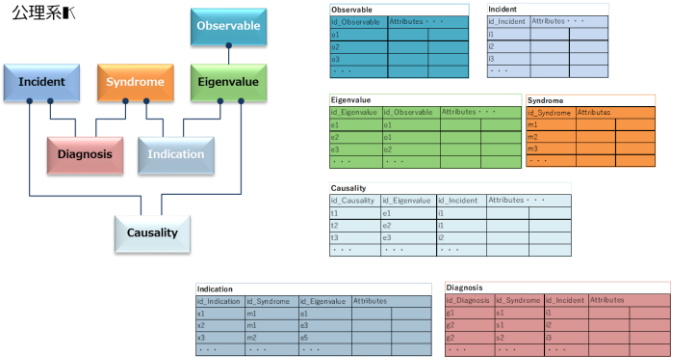


公理系 ◧ は公理系 ◨ を包含している。Syndrome, Diagnosis, Indication が追加されて、より知識が「豊かに」なったといえる。

この場合も、推論の本質部分である、Eigenvalue と Incident が Causality により関係づけられていることには変わりがないが、より多様な推論の設定が可能となる。



公理系 ◧ の公理系 ◨ への付加部分をグラフ表現したエンティティは、同様に表の表現をとれるので、下図のような表を設定することができ。この表現により、SQL による知識の取り出しが可能となりる。



4. 結論

症状・診断型のエキスパート・システムは、公理系 ◨ によりメタ知識を定義することにより、◨ 上の一階述語論理の延長上に ES 推論のメタアルゴリズムを定義できる。

推論の基点と終点により、意味的には前向き、後向きの推論が定義できる。さらに、公理系 ◧ により、より豊饒な体系を構築できる。

公理系 ◨, ◧ はともに、データモデルによるグラフ表示ができる。データモデルを用いれば、entity クラスとそのリレーションシップが、メタ知識であり、そのインスタンスが知識であると考えることができる。

データモデルの表現の一つに表がある。ES 推論は SQE により記述できるので、表として表現された知識の上では、推論は SQL により記述可能である。

以上のような考えに基づくと、エキスパート・システムは、データモデル ◨ ないしデータモデル ◧ にインスタンスを実装して、SQL による推論により実現できると、結論付けられる。

さらに、公理系 ◧ において Symptom の定義に、Eigenvalue Probability 値を導入すれば、推論の二階述語論理化が可能と考えられる。二階の述語論理は一階の述語論理により記述することができるので、この場合でも、SQL による実装が可能であることは明らかである。

以上