高階智能系統

RAISE 理論：打造漸進式、高效使用資料、

結構透明的 AI 系統

（一）高階智能系統理論

R.K.C

第一版 – 預印本

有關本書所述理論的補充內容與討論，請造訪

https://raise-theory.com

或寄送電子郵件至

rkc@raise-theory.com

一版序

本書是《高階智能系統（一）高階智能系統理論》，闡述了高階智能系統理論的邊界、重要概念與工具、演變及未來展望。

「高階智能系統理論」中的「高階」，意味著以容易理解的高層次架構來描述智能系統的運作，它與「低階」方法之間，除了視角不同，也各有擅長。本書一版序撰寫的當下，高階方法與低階方法各自都面臨到一些無法解決的難題，必須兼容並濟、交替使用，才能解釋、架構出符合期待的智能系統。

「智能」指的是「智慧」與「能力」，相較「智慧」一詞，更完整地涵蓋本書所欲描述的一類系統的特性。畢竟，系統若僅具有智慧，但不與外界發生可觀察的互動，那麼，其能力再如何優秀，也無法為我們所用。

智能系統的「系統」定義，則是在發展本書理論的過程中創造出來的，與其他學科當中的此詞定義或有異同。

綜合來說，智能系統指的是這類系統：他們藉由感官取得輸入、根據經驗進行推論、經由學習發生可持續的改變、且行動以嘗試達成意圖。「高階」指的不是這類系統中的一個子類，而是對於這類系統的一個研究方向，本書書名的副標更有助於這種正確理解，至於主標命名為高階智能系統，是為了簡潔描述。

－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－

對於智能系統的研究無疑是相當重要的。人類文明既然是由屬於這類系統的個體組成，若是針對智能系統的架構、表現、缺陷有更深的認識，不只能夠作為認識自身的一種媒介，也能帶來完善人造智能系統的契機。

直到本書一版序撰寫的當下，產學界對於人造智能系統的研究已專注在低階方法上許久，這無疑是高階方法在更往前的一個年代當中發展遲滯，而低階方法先找到了突破口所致。然而，低階方法的缺點顯而易見，找到可補上其低效或不實用處的另一方法體系，已經是現下無可避免的要務。

高階及低階方法之間並無優劣，而各有專注解決的任務及長處。高階方法不處理諸如光線、音波、溫壓等轉換為認知事實的感官過程，也無法以其本身的工具解釋低階方法為何能夠展現出相當高水準的智能表述；同時，低階方法當前面臨到如「學習一項任務造成它項任務表現下滑」、「僅依靠少量輸入難以實現

學習」等問題，且以其邊界內的工具，同樣無法解釋許多包含人類在內，智能系統思考過程中的重要表徵。

尤有甚者，低階方法使用到的數學複雜，注定只能被少數人所理解並掌握。即使忽略所有其餘缺點不談，更假設它短期內突破了所有重要瓶頸，實現了人類等級或超人等級的智能，這種晦澀的特性仍非社會所樂見。

－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－

本書一版所描述的高階理論及方法，承接的皆是多年以來已成熟的方法，而在其上再有發揮。

另外，本書內容亦非著眼於學術上的突破，而在提供當今實現人造智能系統架構上的另一思路。

突破非突破、學術重要性如何，全非其欲所爭。

高階智能系統理論框架發展仍處萌芽階段，期待所有領域先進指教、批評，以及有志之士投入改進。希望十年之間，此書篇幅增加五倍之多、高階方法發展蓬勃，且於社會有益。

謹致

初落筆於　　二零二六年 杏月

一版序定稿於　二零二六年 長夏

本書著成於　　二零二六年 蘭月

R.K.C

目錄

序章：高階智能系統理論的發展

高階智能系統理論的發展始於 R.K.C 於 2018 至 2019 年間，對於「語句中下一個字元的預測可能性」，以及不同自然語言之間「概念」對應的研究。在此之前，R.K.C 在 2011 至 2012 年曾短暫接觸過規則式的 AI 系統設計，並認識到其限制。2013 到 2017 間，R.K.C 維持了對於人工智慧系統設計的興趣，但並未著手於相關理論開發。

2018 年起的研究，最初目的是為改善當時人工智慧翻譯模型的表現，而欲提出一個與機器學習方法不同的功能實現路徑。

－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－

2018 年至 2019 年間，R.K.C 觀察到智能系統的「記憶」功能與「對於實體的物件式理解」，並提出體系中應該存在 T 系統。

同時，即使明確認知到智能並非一定以語言使用為前提，R.K.C 仍然指出「不使用語言就無法描述智能架構」，而確立高階理論以語言理解與表述為中心的核心精神，並提出 L 系統以連結智能系統的語言使用及認知狀態。

由於推論過程廣泛被認為是智能系統的內在過程之一，R.K.C 在 LIT 架構中加入了第三個部分 I 系統，作為驅動認知狀態內部演變的核心，他並初步決定以 LIT 系統架構作為理論進一步發展的出發點。

2020 年迎來了理論發展的重要契機，除了明確真值語句的中心地位外，這段期間 R.K.C 也發展了一系列的工具，如推論矩陣等，使 LIT 系統的實踐方法得到更加清晰的描述。

此間最為關鍵的一步是在觀察到真值語句間普遍存在的矛盾後，為真值設立了效力邊界，出現「域」的概念。

R.K.C 此時也認知到域是整個理論框架的轉捩點，從符號與工具的設計，跨向對於智能系統本質的描述，顯著拓展了此理論框架的任務範疇。

－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－

2021 年標誌了另一系列的關鍵突破。

除了使用注意力域及意圖來初步解釋智能系統的決策與行動以外，這時「系

統」的定義在理論當中也漸趨明確。系統與域的兩面性也在這段時間得到闡述。

2022 年的關注重點在於同構。

為了解釋智能系統何以在不同的結構之下仍顯現出相同或相似的表現，以及為人工智能系統理論設定清晰目標，系統的構造、同構、解釋力與可解釋性的概念在此間陸續被提出。

另外，這時 R.K.C 也探討了系統及域與其內外部的關係，並提出演序的概念。與此同時，探討域內及域間的矛盾消解過程成為學習理論的開端，範式的概念被用來解釋經驗與學習帶來的重複性推論與行動過程（後來將此觀念使用「定形」與「定式」稱呼，並將「範式」一詞的指涉改為如本書中所述）。

－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－

2023 年至 2024 年，R.K.C 的關注重點在生活的其他層面，這段時間，低階方法開始受到廣泛關注，其表現也迅速進步，此間低階方法短期內似乎看不到發展瓶頸。

至 2025 年，R.K.C 重新整理了高階方法的各個理論範疇，並檢視了高階理論在與低階方法間協作可能性，以及評估高階方法是否有助解決某些低階方法越發明顯的缺點。理論發展層面，這段時間 R.K.C 提出了工具標記 @ 及學習理論的收斂方式：「穩定度」。

2026 年，數個方向上的進一步發展促使 R.K.C 將整個理論架構重新編排表述，並準備向外提出。一是不完全系統及推論距離的概念出現，解決了長時間困擾整個理論框架的「無限推論」問題，二是高階理論的概念使理論框架與低階方法間有了乾淨的區隔（在此之前，高階理論是 R.K.C 所欲發展的「另一個理論路線」，並未明確的以「高階」為其出發點或核心），三是後撤堆疊使學習理論越發完善。

本書第一版於 2026 年春夏之際動筆。

－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－

高階智能系統理論的發展並非一條直線，過程中開發出的多數概念，之後已棄而不用，許多工具與認知，也為更貼近本質、更為優美的設計所取代。

高階智能系統理論更將不是一家之言，就如同理論中的重要概念之一，「同構」，許多設計即使外觀大異其趣，其功能卻並無二致。究其根本，能正確描述

智能系統行為，且具可解釋性、優美、適於理解、可以實踐的理論，即是優秀的智能系統理論。

拋磚引玉，是期待遍野繁花。

－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－

高階智能系統理論關注重點及發展時間表（僅列重要發展）：

2018～2019	• 預測語句中下一個字的方法 • LIT 架構
2020	• 真值語句 • 推論矩陣 • 域
2021	• 注意力 • 意圖與行動 • 系統 • 域的推移
2022	• 同構 • 解釋力 • 演序 • 範式 • 矛盾消解
2023～2024	（無）
2025	• 穩定度 • 高階理論與大型語言模型間的協作 • @標記
2026	• 不完全系統 • 高階的概念 • 推論距離 • 後撤堆疊

第一部分 —— 完全系統的理論基礎

第一章：L 系統

1.1 點號標記

點號標記將句子以點號（.）分隔為數個較小的部分，通常是分成複數個詞組。分隔詞組時，並不需要謹慎考慮點號放置的位置，考量到點號標記的實際作用方式，即使點號放置的位置不同，也都能夠發揮效用。

對幾個簡單的句子進行點號標記的結果如下：

- 蘋果.是.紅色.的
- 山.上.的.空氣.很.清新
- 小明.想.玩.手機.遊戲.，.但.他.決定.先.把.功課.做完

由上面的幾個例子所見，點號標記與一般直覺、習慣上的分詞方法大致相符，但是也會遇到模稜兩可處。比如「山上」應該是不被分隔的一個詞，還是如範例中寫成「山.上」呢？「做完」應該如範例中兩字不被分隔，還是應該考慮到「做」本身的動詞角色，寫成「做.完」呢？

事實上，分詞方式本身具有不確定性。在語言使用者的角度，分隔與不分隔都是「合理」且「可理解」的，並且分隔與不分隔都能產生有效的認知結果。

因此，我們可優先考慮工具操作所需，對句子進行任意合理的點號標記。

－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－

諸如逗號、頓號、句號、引號、書名號等各種標點符號，我們在其前後都進行點號標記，例如：

- 他.問.了.妹妹.：.「.現在.幾點.？.」
- 市場.裡.有.花卉.、.水果.、.碗盤.、.瓷器.等.各種.商品.。
- 圖書館.裡.有.他.最.喜歡.的.書. .–. .《.演算法.生存.指南.》.。

上面第三個例子中，可以看到句子裡出現空格的情形。如果在進行點號標記前，是否有該空格在句中都不會改變句子的意思，那麼我們就可以任意選擇在空格前後加上點號，或者直接忽略該空格，比如也可將其寫成：

- 圖書館.裡.有.他.最.喜歡.的.書.–.《.演算法.生存.指南.》.。

1.2 斜線標記

斜線標記是在詞組前方加上數字編號與斜線符號，比如「1/」、「2/」。

以下的例子中，我們可以看到斜線標記的用途：

蘋果.是.紅色.的.。

- 1/水果.是.紅色.的.。
- 蘋果.是.1/顏色.的.。
- 1/水果.是.2/顏色.的.。

上面三個含有斜線標記的寫法，都是原句使用斜線標記後的「合法（符合規範）」表示。

這裡，我們可以看出因為「蘋果是一種水果」，所以可將蘋果寫成「1/水果」；而「紅色是一種顏色」，因此可以將紅色寫成「1/顏色」。當單一個句子裡有複數個以斜線標記形式出現的詞組時，我們預設使數字以它們的出現順序遞增。

使用「數字」做標記以及規定其「遞增」，不代表一般的智能系統對於語言產生認知時，過程中實際有這樣的步驟發生。這只是方便操作的一種工具設計。

另外，諸如「蘋果是一種水果」這樣的句子，不一定對於每個人都成立。就像「蜂蛹是一種食物」這樣的認知，對於一些人來說是理所當然，對於其他人來說則不是，不過，這不會影響到此標記方式搭配其他概念的使用。

－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－

進行斜線標記時，「多個被點號標記所分隔的詞組」可以被納入「同一個斜線詞組」當中。

研究.古希臘文.的.學者.們.都.聚集.在.這裡

1/學者.們.都.聚集.在.這裡

1/學者.們.都.聚集.在.2/地點

上面標記隱含的前提是：「研究.古希臘文.的.學者」是「學者」，或者說，是「學者」這個概念可代指的對象之一。

－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－

他.喜歡.去.走訪.暢銷.旅遊.書.上.列出.的.熱門.景點

1/人.喜歡.去.走訪.2/書.上.列出.的.3/景點

1/人.喜歡.去.走訪.2/書.上.列出.的.3/地點

1/人.喜歡.去.2/動作

1/人.喜歡.2/動作

像這個例子所示，一個詞組可以經過多次的斜線標記，如「熱門.景點」變為「3/景點」，再變為「3/地點」。也可以透過將一個句子中多個分隔的詞組多次進行斜線標記，來漸次得到分隔詞組數較少的句子。

斜線標記使用的類別名稱，即斜線後方跟隨的詞並沒有特定限制，也沒有一個固定參考列表，使用的詞是什麼，只影響到進行推論後得到的結果，而我們可從「是否能夠得到正確推論」來判斷是否使用了「正確的標記方式」，更多細節請見推論相關章節。

另外，當斜線標記使用的類別名稱本身是由多個詞所構成的時候，為了避免與類別名稱以外的部分混淆，我們改以「_」來分隔類別名稱內的詞組，比如「/高中_一_年級_的_學生」。放在一個句子內，形式如：

1/高中_一_年級_的_學生.應該.已經.學會.國中.階段.的.知識

注意到，這與：

1/高中.一.年級.的.學生.應該.已經.學會.國中.階段.的.知識

是不同的標記方式，前者如果將「小明」代入，會成為：

小明.應該.已經.學會.國中.階段.的.知識

後者如果將「RT 高中」（某一所假設中的高中名稱），則會成為：

RT.高中.一.年級.的.學生.應該.已經.學會.國中.階段.的.知識

1.3 錢號標記

值得注意的是，在某些語言（如部分東亞及東北亞語言等）中，助動詞與動詞等不隨著人稱變位，因此在高階智能系統理論的發展初期，並沒有使用到錢號「$」標記。

然而，為了使 L 系統的標記方法在一些其它語言當中也能夠適用，必須加上額外的表記工具：

- He likes to play computer games.
- They like to watch anime.
- She is a cat lover, while he likes dogs more.

使用點好標記：

- He.likes.to.play.computer.games...
- They.like.to.watch.anime...
- She.is.a.cat.lover.,. .while.he.likes.dogs.more...

如上所示，僅使用到點號標記時，不會一併使用到錢號標記。

如下方的例子中所示，由於「1/person」這樣的斜線標記可以代入諸如「I」、「you」、「he」、「she」、「her elder sister」等，而會影響到後方「like」一詞應寫為「like」或「likes」，因此我們在其後方加上「$1」，成為「like$1」，表示這個詞組在「1/person」代入實例後的正確寫法，應視「1/person」的實例為何而定。

- 1/person.like$1.to.play.computer.games.
- 1/persons.like$1.to. watch.anime...
- 1/person.be$1.cat.lover.,. .while.2/person.like$2.dogs.more...

上例中，「be$1」在「1/person」代入實例後的正確寫法，跟隨「1/person」代入的實例，「like$2」在「2/person」代入實例後的正確寫法，跟隨「2/person」代入的實例而定。

另外，同樣以「_」來分隔類別名稱內的詞組，且可以多次進行斜線標記。

- 1/person.likes.to.play.2/games...
- 1/persons.like.to.2/act...
- 1/person.is.2/type_of_person.,. .while.3/person.likes.4/things.more...

- 1/person.likes.to.2/act...
- 1/persons.like.to.2/act...

- 1/person.is.2/type_of_person.,. .while.3/person.4/act...

- 1/person.like$1.to.2/act...
- 1/persons.like$1.to.2/act...
- 1/person.be$1.2/type_of_person.,. .while.3/person.like$3.4/things.more...

1.4 L 系統的功能

L 系統的主要功能是將「所有的合法語句」歸納為「有限種的分類」，使得 I 系統中不需要存在無限多種規則。

點號標記將「語元」（語句的意義單位）劃分出來，並使斜線標記在應用上不會與原句內容產生混淆。

經過多次斜線標記的縮短後，多個長句可以被縮短為同樣表示方式的「含點號及斜線標記的短句」，使得在 I 系統中，它們可以被適用在同樣的規則上。更多細節，請見 I 系統的介紹。

1.5 真值語句

「真值」的概念在高階智能系統理論中非常重要。有別於命題邏輯、述詞邏輯當中注重判斷語句具有「真」和「假」兩種值中的哪一個，高階智能理論則更加關注一個語句（敘述）是否「具有真值」。

語句可能具有真值而成為「真值語句」，或者不具真值而「不是真值語句」，比如：

公雞.有.兩.隻.腳

（在一般人的認知當中，具有真值）

公雞.有.四.隻.腳

（在一般人的認知當中，不具有真值）

習慣上，使用大寫英文字母 A、B、C… 來標示語句。

－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－

當一個「語句 A」的反面「語句 B」是真值語句時，語句 A 就具有「假」值。我們將語句 A 的反面寫作「～A」，因此當「語句 A」的反面「語句 B」有真值時，語句 A 就具有「假」值，使得「～A」有真值。

這也就是說，我們可以透過一個語句的反面是否具有真值，來判斷該語句本身是否具有假值。

玫瑰.是.一種.水果　　A
（在一般人的認知當中，不具有真值）

玫瑰.不是.一種.水果　　B
（在一般人的認知當中，具有真值）

～(玫瑰.是.一種.水果)　　～A
（在一般人的認知當中，具有真值）

其中，「～(玫瑰.是.一種.水果)」指的是「玫瑰.是.一種.水果」的反面。

當然，一個語句是否具有真值，時常需要考慮很多因素。比如「玫瑰是一隻動物」一般來說並不是真的，但有可能有人將其寵物狗取名叫做玫瑰，這時「玫瑰是一隻動物」就會在有限的範圍內具有真值。關於真值效力的範圍，請見「域」章節中的深入討論。

第二章：I 系統

2.1 真值語句的展開

許多語句涵蓋了多個概念，是多個語句的複合。這類語句的真值會與多個「含有較單純概念的語句」的真值相關，比如：

小明.喜歡.讀書.，.也.喜歡.運動.。　A

小明.喜歡.讀書.。　B

小明.喜歡.運動.。　C

語句 A 涵蓋了語句 B 與 C 的意思，使得如果 A 是真的，B 與 C 都必定是真的；而若 B 與 C 任一個不是真的，A 也不是真的。

－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－

他.把.籃球.舉高.到.頭.上.。　D

他.把.籃球.舉高.。　E

E 語句是 D 語句所表達的概念的一部分，因此當 D 語句是真值語句，E 也是真值語句。

植物.在.白天.吸收.二氧化碳.，.但是.晚上.時.吸收.氧氣.。　F

植物.在.白天.吸收.二氧化碳.。　G

植物.在.晚上.時.吸收.氧氣.。　H

F 語句涵蓋了 G、H 兩個語句的意思。

另外，以 H 為例，有其他表示方式的意思與它相同：

晚上.時.，.植物.會.吸收.氧氣.。　　I

植物.晚上.時.會.吸收.氧氣.。　　J

像這樣，透過分析語句表達的概念之間是否具有包含、複合的關係，或者一個語句是否是另一語句的同義表達方式，我們可以將一個語句的真值擴展到其他語句上。

－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－

也就是說，如果「A 為真，B 就為真」的規則存在，那麼 A 是真值語句時，B 也是真值語句；如果「A 為真，那麼 B、C、D、… 等 10 個語句都為真」的規則存在，那麼我們只需關注 A 是否為真即可，不需一一檢視其它每個同義的表示方式，也不需個別檢視「只含有 A 的一部分概念」的語句。

像這樣從單一語句的真值擴展到複數語句的真值的過程，稱為真值語句的「展開」，展開會將真值套用到任何其它「真值總是跟著原語句 A 的真值變動」的語句上。

展開的概念可以幫助我們簡化真值的分析，也使「推論」的結果能夠充分向外擴展。

2.2 規則與規則的表記

規則可以使用真值語句表示，並且描述了真值語句間的關係。

如果.明天.下雨.，.小明.就.不.出門。　　A

明天.下雨.。　　B

小明.明天.不.出門。　　C

語句 A 是一個真值語句，且描述了語句 B、C 之間的關係。至於為何 C 語句中也需要帶有「明天」一詞、A 語句何時成立等問題，請見後續章節討論。

語句 B 與 C 之間的關係是：

- 當 B 是「真的」的時候，C 也是「真的」。
- 當 B 不是「真的」的時候，C 有可能是「真的」，也可能不是「真的」。

這樣的語句關係，在命題邏輯中，可以表示為 B -> C，符號「->」是箭頭符號，讀作「B 則 C」，或「若 B，則 C」。

在高階智能理論中，「->」（則）的定義亦可以使用上面對於語句 B 和 C 之間關係的敘述來理解。

我們另外加入一種表記方式：

A.=.B->C

在這裡，「.=.」的意義與邏輯學中的「<->」基本相同，也就是 A -> (B->C)，且 (B->C) -> A，「.=.」只有在前後都是語句代號或語句代號之間的邏輯關係時，才具有這個意思。

至於「3.+.3.=.6」或「質量.=.體積.x.密度」這些語句內，雖然也出現「.=.」，但代表的是句中有「=」，又經過點號標記處理。

－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－

除了形如 A -> B、B -> C 的這樣的簡單規則外，也會有一些較為複雜的規則，它們是更簡單的規則間的合成。

如果.明天.是.禮拜一.而且.不.是.當.月.第.三.週.，.這.家.餐廳.就.不.開門。	D
明天.是.禮拜一.。	E
明天.是.當.月.第.三.週.的.其中.一天。	F
這.家.餐廳.不.開門。	G

上面語句 D 至 G 之間的邏輯關係是：「若 E 且非 F，則 G」，且因為 D 描述的就是「若 E 且非 F，則 G」，所以「D 則（E 且非 F，則 G）」，且「（E 且非 F，則 G）則 D」。

再看一個例子。

要是.小明.不.來.或.小美.不.來.，.我們.就.不.去.唱歌.了.。　　J

小明.不.來.。　　K

小美.不.來.。　　L

我們.不.去.唱歌。　　M

上面語句 J 至 M 之間的邏輯關係是：「若 K 或 L，則 M」，且因為 J 描述的就是「若 K 或 L，則 M」，所以「J 則（K 或 L，則 M）」，且「（K 或 L，則 M）則 J」。

另外，語句編號不使用英文字母「I」及「O」，以避免與數字「1」和「0」混淆，也不使用英文字母「X」，以避免與運算符號「乘號」混淆。

－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－

為了精簡地表達如上面例子的一些比較複雜的邏輯關係，我們使用到其它的一些命題邏輯的符號。需注意，括號內的運算優先進行：

符號示例一：

寫作	讀作	解釋
A^B	A 且 B	A 是真的，B 也是真的
AvB	A 或 B	A 是真的，或者 B 是真的，也可能 A 和 B 都是真的
(A^B)vC	A 且 B，或 C	「A 且 B」是真的，或者 C 是真的，也可能「A 且 B」和 C 都是真的

符號示例二（使用「～」而非其它傳統上的邏輯符號來表示「非」，是為了打字簡便）：

寫作	讀作	解釋
～A	非 A	A 不是真的
～（A^B）	非「A 且 B」	「A 且 B」不是真的。 代表 A 不是真的，或者 B 不是真的，或者 A 和 B 都不是真的。 也就是說，不可能 A 是真的，而且同時 B 也是真的。
～（AvB）	非「A 或 B」	「A 或 B」不是真的。 代表 A 不是真的，而且同時 B 也不是真的。

－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－

我們使用如 R_1、R_2、…、R_n 的符號來代指規則。在 R_1.=.A->B 這樣的規則中，A 是前提、B 是推論結果；在 R_2.=.C^D->E 這樣的規則中，C、D 都是前提、E 是推論結果；在 R_3.=.F->G^H 這樣的規則中，F 是前提、G、H 都是推論結果。

蘋果西打.特價	A
1/物.特價	A’
蘋果西打.比.原本.便宜	B
1/物.比.原本.便宜	B’
蘋果西打.特價 -> 蘋果西打.比.原本.便宜	R_1
1/物.特價 -> 1/物.比.原本.便宜	R_1’

上例中，語句 A 經過任意次斜線標記後成為 A’，而規則 R_1 以 A’ 為前提，那麼可以將 A 中的語元「代入」 A’，使 A’ 中經斜線標記的語元「還原」為 A 中的語元。

類似的，如果規則的推論結果中有 B’，而語句 B 可經過任意次斜線標記後成為 B’，那麼可以將 B 中的語元「代入」 B’，使 B’ 中經斜線標記的語元「還原」為 B 中的語元。

當規則中所有經過斜線標記的語元都被代入而還原後，例如 R_1’ 還原為 R_1，規則就由一個具有一定抽象性的原則成為一個實例。要注意的是，無論是實例還是抽象性的規則，都可能有真值，也可能沒有真值，也就是說，可能成立也可能不成立。

任何合法的代入過程，都會使一個有真值（成立）的規則，在其所有的前提都有真值的情況下，推論出同樣有真值的結果。

如果觀察到代入產生的實例，會使得有真值的規則推論出無真值的結果，可能是發生了「語元階層設計不正確」、「規則本身並不總是有真值」、「擴張錯誤」等問題，請見 T 系統、域、錯誤等各章節的討論。

2.3 推論矩陣

推論矩陣是高階智能理論中進行推論的基礎工具。

最基本的表達方式是由一個或複數個語句的真值，來得到另一個或另一些語句的真值，比如：

If	1/人.在.2/高中.工作	D
	1/人.是.助教	E
->	1/N.是.高中.助教	F

注意到，在一個推論矩陣當中，同一個實體或同一個概念中介使用同一個編號，比如：

If	1/人.是.2/人.的.哥哥	G
->	2/人.是.1/人.的.弟弟	H

像這樣，任何使 G 語句成立的「1/人.和.2/人」，都可套用到 H 當中，使 H

（2/人.是.1/人.的.弟弟）成立。

－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－

推論矩陣也可用來表示真值語句的展開：

If	1/人.既.活潑.又.聰明	J
->	1/人.很.活潑	K
	1/人.很.聰明	L

注意到，推論符號「->」的上方及下方（下方包含同一行）一定有一或多個語句，「If」只是用來標誌一個推論矩陣的開始。在「->」上方的語句是規則的前提，當「->」上方的語句都具有真值時，「->」下方的語句（推論結果）也都有真值。

－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－

再看到另一個例子：

[規則 R_3]

If	1/三角形.的.三.個.角.是.2/角.、.3/角.，.和.4/角	M’
	2/角.是.5/數字.度	N’
	3/角.是.6/數字.度	P’
	180.-.5/數字.-.6/數字.=.7/數字	Q’
->	4/角.是.7/數字.度	S’

規則 R_3 指出了三角形的三個內角角度和的關係。其中一個這個規則的實例是：

If	三角形.ABC.的.三.個.角.是.角.A.、.角.B.，.和.角.C	M
	角.A.是.20.度	N
	角.B.是.25.度	P
	180.-.20.-.25.=.135	Q
->	角.C.是.85.度	S

這裡隱含了「三角形.ABC」是一種三角形，因此可以寫成「1/三角形」；「角.A」是一個角，因此可以寫成「2/角」;「20」是一個數字，因此可以寫成「5/數字」等。當這些都成立時，就可以根據規則 R_3，來使用任意三角形中任意兩個角的度數來得到另一個角的度數了。

－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－

表達「1/三角形.的.三.個.角.是.2/角.、.3/角.，.和.4/角」（M’）這種意思的語句有很多種表達方式，如「1/三角形.有.2/角.、.3/角.，.和.4/角.三.個.角」（T’）、「1/角.、.2/角.，.和.3/角.是. 4/三角形.的.三.個.角」（U’）等。

因此，為了讓其它表達方式也都可以套用到規則 R_3 當中，而不需刻意另外建立以 T’、N’、P’、Q’ 作為前提得到 S’ 的規則，又再建立以 U’、N’、P’、Q’ 得到 S’ 的規則等等，我們可以轉而建立如下規則：

If　1/三角形.有. 2/角.、.3/角.，.和.4/角.三.個.角　　T’

->　1/三角形.的.三.個.角.是.2/角.、.3/角.，.和.4/角　　M’

以及（這裡為了方便理解，未按照規範順序編號）

If　2/角.、.3/角.，.和.4/角.是.1/三角形.的.三.個.角　　U’

->　1/三角形.的.三.個.角.是.2/角.、.3/角.，.和.4/角　　M’

這樣一來，只要經過兩次規則套用，比如先從 T’ 得到 M’（因為 T’->M’），或者從 U’ 得到 M’ （因為 U’->M’），然後再套用規則 R_3，就可以實現一樣的內角角度推論過程。

2.4 推論鏈

推論鏈是指達到目標真值前的一系列推論過程。

考慮這個問題的幾種常見答案：

▶ 步驟（1）

「日本和新加坡，那一個地方更容易下雪？」

- 日本比新加坡容易下雪
- 新加坡比日本容易下雪

- 日本和新加坡一樣容易下雪
- 有時日本比較容易下雪，有時新加坡比較容易下雪
- 日本和新加坡都不下雪

－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－

在不同的條件下，可能會有不同的答案成立。我們要找到是否有其中一或多個語句成立，或者沒有語句成立時，可以進行以下的過程：

步驟（1）：列出可能的答案，如上方所示

步驟（2）：對可能的答案（語句）進行點號標記與任意次斜線標記

步驟（3）：找到推論結果（在推論矩陣中，位於「->」的右側或下方）中，含有（2）中的某個語句的所有規則

步驟（4）：列出所有（3）中找到的規則的前提（位於「->」上方的語句）

步驟（5）：根據（4），確認是否有一些（3）中找到的規則，它們的所有前提皆為真

步驟（6）：如果（5）中找不到任何規則，再確認是否有一些（3）中找到的規則，它們的前提中除了已經有真值的那些以外，其它所有的前提的「反面」也沒有真值

步驟（7）：對於（6）中找到的規則中，尚無真值的前提，進行（2）到（6）步驟，直到在步驟（5）找到一個以上的規則，或者在（6）中無法再找到任何規則

上方的過程，叫做建構「逆推論鏈」，因為它是從欲找到的結果倒推，看是否有「足夠的真值」能夠讓「至少一個規則」推論出「欲找到的結果的真值」。

－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－

從「日本比新加坡容易下雪」出發，先寫成「日本.比.新加坡.容易.下雪」。

如果這個語句本身就有真值，那可以直接得到它是一個真值語句。但我們在此先不要假設它已經有真值，也不要假設它的反面已經有真值，以做推論的示範。

- 步驟（2）

將「日本.比.新加坡.容易.下雪」寫成任意次斜線標記的結果：

1/地方.比.新加坡.容易.下雪

日本.比.1/地方.容易.下雪

日本.比.新加坡.容易.1/動作

1/地方.比.2/地方.容易.下雪

1/地方.比.新加坡.容易. 2/動作

日本.比.1/地方.容易. 2/動作

1/地方.比.2/地方.容易. 3/動作

1/地方.比.2/地方. 3/容易_動作

上面「/容易_動作」只是任意選定的一個類別名稱。

－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－

- 步驟（3）

接下來，針對任意選定的「1/地方.比.2/地方.容易.下雪」，尋找是否有規則使得這個語句在其推論結果之中。

這裡我們直接假設描述「越.冷.的.地方.越.容易.下雪」的 R_1 成立。A’ 和 B’ 有「’」符號，是因為它們是使用到斜線標記的語句，當它們的斜線標記部分都代入語元而還原成實例時，就寫為沒有「’」符號的 A 和 B。

[R_1]

If　1/地方.比.2/地方.冷　　A’

-> 1/地方.比.2/地方.容易.下雪　　B’

- 步驟（4）

規則 R_1 的前提是 A’。

- 步驟（5）

此時，只要 A（日本.比.新加坡.冷）成立，就可以推論出 B（日本.比.新加坡.容易.下雪）。

在此，我們先不直接假設 A 成立，以繼續示範逆推論鏈的構造過程。

因為不直接假設 A 成立，所以步驟（5）中找不到任何規則。

- 步驟（6）

因為（5）中找不到任何規則，而（3）中找到的規則 R_1，它的前提中除了已經有真值者（這裡正好沒有）外，A（日本.比.新加坡.冷）沒有真值，而 A 的反面 ～A=～(日本.比.新加坡.冷) 也沒有真值，所以我們繼續進行步驟（7）。

- 步驟（7）

步驟（6）中找到的規則中，尚無真值的前提是 A （日本.比.新加坡.冷），我們繼續針對 A 進行（2）到（6）步驟。

－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－

- 步驟（2）

將「日本.比.新加坡.冷」寫成任意次斜線標記的結果：

1/地方.比.新加坡.冷

日本.比.1/地方.冷

日本.比.新加坡.1/狀態描述

1/地方.比.2/地方.冷

1/地方.比.新加坡.2/狀態描述

日本.比.1/地方.2/狀態描述

1/地方.比.2/地方.3/狀態描述

- 步驟（3）

接下來，針對任意選定的「1/地方.比.2/地方.冷」，尋找是否有規則使得這個語句在其推論結果之中。

這裡我們直接假設描述「緯度.越.高.的.地方.越.冷」的 R_2 成立。

[R_2]

If　1/地方.比.2/地方.的.緯度.高　　C’

-> 1/地方.比.2/地方.冷　　D’

- 步驟（4）

規則 R_2 的前提是 C’。

- 步驟（5）

此時，只要 C（日本.比.新加坡.的.緯度.高）成立，就可以推論出 D（日本.比.新加坡.冷）。

在此，我們先不直接假設 C 成立，以繼續示範逆推論鏈的構造過程。

因為不直接假設 C 成立，所以步驟（5）中找不到任何規則。

- 步驟（6）

因為（5）中找不到任何規則，而（3）中找到的規則 R_2，它的前提中除了已經有真值者（這裡正好沒有）外，C（日本.比.新加坡.的.緯度.高）沒有真值，而 C 的反面 ～C=～(日本.比.新加坡.的.緯度.高) 也沒有真值，所以我們繼續進行步驟（7）。

- 步驟（7）

步驟（6）中找到的規則中，尚無真值的前提是 C （日本.比.新加坡.的.緯度.高），我們繼續針對 C 進行（2）到（6）步驟。

－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－

▶ 步驟（2）

將「日本.比.新加坡.的.緯度.高」寫成任意次斜線標記的結果：

1/地方.比.新加坡.的.緯度.高

日本.比.1/地方.的.緯度.高

日本.比.新加坡.的.緯度.1/形容詞

1/地方.比.新加坡.的.緯度.2/形容詞

日本.比.1/地方.的.緯度.2/形容詞

1/地方.比.2/地方.的.緯度.高

1/地方.比.2/地方.的.緯度.3/形容詞

…

▶ 步驟（3）

接下來，針對任意選定的「1/地方.比.2/地方.的.緯度.高」，尋找是否有規則使得這個語句在其推論結果之中。

這裡我們直接假設描述「如果 A 在北半球，B 也在北半球，而且 A 比 B 更北邊，那麼 A 的緯度比 B 高」的 R_3 成立。

[R_3]

If 1/地方.和.2/地方.都.在.北半球 E’

1/地方.比.2/地方.更.北邊 F’

-> 1/地方.比.2/地方.的.緯度.高 G’

▶ 步驟（4）

規則 R_3 的前提是 E’ 和 F’。

- 步驟（5）

此時，只要 E（日本.和.新加坡.都.在.北半球）成立，且 F（日本.比.新加坡.更.北邊）也成立，就可以推論出 G（日本.比.新加坡.的.緯度.高）。

在此，我們直接假設 F 成立，但不直接假設 E 成立，以繼續示範逆推論鏈的構造過程。

因為不直接假設 E 成立，所以步驟（5）中找不到任何規則。

- 步驟（6）

因為（5）中找不到任何規則，而（3）中找到的規則 R_3，它的前提中除了已經有真值者（F）外，E（日本.和.新加坡.都.在.北半球）沒有真值，而 E 的反面～E=～(日本.和.新加坡.都.在.北半球) 也沒有真值，所以我們繼續進行步驟（7）。

- 步驟（7）

步驟（6）中找到的規則中，尚無真值的前提是 E （日本.和.新加坡.都.在.北半球），我們繼續針對 E 進行（2）到（6）步驟。

－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－

- 步驟（2）

將「日本.和.新加坡.都.在.北半球」寫成任意次斜線標記的結果：

1/地方.和.新加坡.都.在.北半球

日本.和.1/地方.都.在.北半球

日本.比.新加坡.都.在.1/半球

1/地方.和.新加坡.都.在.1/半球

日本.和.1/地方.都.在.2/半球

1/地方.和.2/地方.都.在.3/半球

1/地方.比.2/地方.都.3/狀態

…

- 步驟（3）

接下來，針對任意選定的「1/地方.和.2/地方.都.在.3/半球」，尋找是否有規則使得這個語句在其推論結果之中。

這裡我們直接假設描述「如果 A 在某個半球，B 也在某個半球，那麼 A 和 B 都在那個半球」的 R_4 成立。

[R_4]

If　1/地方.在.2/半球　　H’

　　3/地方.在.2/半球　　J’

->　1/地方.和.3/地方.都在.2/半球　　K’

- 步驟（4）

規則 R_4 的前提是 H’ 和 J’。

- 步驟（5）

此時，只要 H（日本.在.北.半球）成立，且 J（新加坡.在.北半球）也成立，就可以推論出 K（日本.和.新加坡.都在北半球）。

在此，我們直接假設 H 與 J 成立，所以步驟（5）中找到規則 R_4。

因為（5）中找到了規則 R_4，所以根據前述推論過程與假設為真的前提，語句「日本.比.新加坡.容易.下雪」為真。

－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－

逆推論鏈：

假設這些真值成立：{ R_1, R_2, F, R_3, J, H, R_4 }，欲求 B

R_1=A’->B’

[R_1] A ->B

(A’ =D’)?

R_2=C’->D’

[R_2] C ->D

(C’=G’)?

R_3=(E’^F’)->G’

F

[R_3] E^F->G

(E’=K’)?

R_4=(H’^J’)->K’

H^J

[R_4] H^J->K

（逆推論鏈完成）

「R_1=A’->B’」、「R_3=(E’^F’)->G’」這類的表述，是將規則本身的內容寫出。

諸如「[R_1] A ->B」、「[R_2] C ->D」這類的表述，意思是「根據 R_1，如果 A，則 B」、「根據 R_2，如果 C，則 D」。

諸如「(A’ =D’)?」這類的表述，它的意思是「A’（正好等於 D’）是有真值的嗎？」，也可以只寫為「(A’)?」，代表「A’ 是有真值的嗎？」。

先寫一行「F」，下一行再寫「[R_3] E^F->G」，意思是「F 已經成立了」，接著，「根據 R_3，如果（E 與 F），則 G」，因為前面提到了 F 已經成立，所以接著是寫「(E’=K’)?」，而不用再寫一行「(F’)?」。

類似的，先寫一行「H^J」，下一行再寫「[R_4] H^J->K」，意思是「H 和 J 已經成

立了」，接著，「根據 R_4，如果（H 與 J），則 K」，因為前面提到了 H 和 J 已經都成立了，所以逆推論鏈的構造在此完成。

－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－

我們可以將逆推論鏈重新寫為正推論鏈，或簡稱為推論鏈：

假設這些真值成立：{ R_4, H, J, R_3, F, R_2, R_1 }，欲求 B

R_4=(H'^J')->K'

H^J

[R_4] H^J->K

K

R_3=(E'^F')->G'

F

K=E

[R_3] E^F->G

G

R_2=C'->D'

G=C

[R_2] C ->D

D

R_1=A'->B'

D=A

[R_1] A ->B

B

（正推論鏈完成）

B 為真，即「日本.比.新加坡.容易.下雪」為真。

H：日本在北半球。

J：新加坡在北半球。

R_4：如果 A 在某個半球，B 也在某個半球，那麼 A 和 B 都在那個半球。

E=K：日本和新加坡都在北半球。

F：日本比新加坡更北邊。

R_3：如果 A 在北半球，B 也在北半球，而且 A 在 B 的北邊，那麼 A 的緯度比 B 高。

C=G：日本比新加坡的緯度高。

R_2：緯度越高的地方越冷。

A=D：日本比新加坡冷。

B：日本比新加坡容易下雪。

R_1：越冷的地方越容易下雪。

逆推論鏈描述了智能系統調用已認知到的真值來決定「給定目標範圍」、「未知真值」的語句是否有真值的方法。

智能系統可以同時使用正推論與逆推論。逆推論更能保證思考過程完成既定的任務；正推論則會使真值往外展開，但是無法確保是否會展開到當前任務目標含有的一系列語句上。

有時候，逆推論因為搜尋空間可能很大（需要找到並針對推論結果中含有待定真值的規則進行計算），可能會過度耗能，而必須與正推論同時使用已完成任務。

此時，可以考慮先列出一定範圍內（先理解為看起來較為相關）的真值語句，（上例中，比如句中同時有「日本」和「新加坡」的真值語句，或者有「下雪」、「冷」等關鍵概念的真值語句），再選取前提也在同範圍內的規則，（比如前提中有「日本」、「新加坡」、「/地方」、「/國家」的規則，或者有「下雪」、「冷」等關鍵概念的規則），並根據可用資源，進行有限步驟的推論，以期得到一些逆推論過程中尚待推論出的真值，以增加給定時間內完成整個推論過程的可能性。

第三章：T 系統與 LIT 架構

3.1 實體與概念中介

真值語句中的任何部分，都屬於實體或概念中介。

經過點號標記的真值語句當中，實體或概念中介都可能會跨過數個點號，或者由多個不相連的「點號之間的部分」組成，但無論實體還是概念中介，都不會在「沒有直接接在某個點號後面」的位置開始，也不會在「沒有直接接在某個點號前面」的位置結束。也就是說，它們的開始處與結尾處必定都會涵蓋點號與點號之間的某個完整詞組。

小明.昨天.買.的.那.台.特斯拉.開.在.路.上.很.拉風

以下範例中，中括號（[]）裡的內容是實體：

正確：[小明].昨天.買.的.那.台.[特斯拉].是.[四輪傳動.電動車]

正確：[小明.昨天.買.的.那.台.特斯拉].是.四輪傳動.[電動車]

正確：[小明.昨天.買.的.那.台.特斯拉].是.[四.輪.傳動.電動車]

錯誤 1：[小明].昨天.買.的.那.台.[特斯拉].是.四輪[傳動.電動車]

錯誤 2：[小明.昨天.買.的.那.台.特斯拉].是.[四.輪].傳動.電動車

上方的範例當中，錯誤 1 試圖將「傳動電動車」視為一個實體，但實體的開頭處是在被點號分隔開的一個詞組中間，而非開始處。錯誤 2 將「四輪」視為一個實體，忽略了它是用以修飾其後的「傳動電動車」。

同時，由正確的例子中，可以看到：

（1）語句中對實體的標記並非總是只有一個正確的方式

（2）實體與「名詞」概念的區別在於，「特斯拉」這樣的單詞既是名詞，也是實體，但是「小明.昨天.買.的.那.台.特斯拉」、「那.台.特斯拉」等屬於實體，但一般而言，不會被看作是單一名詞。

從語句中分離出實體後，其餘的部分就是一或多個概念中介。概念中介描述了實體之間的互動關係、實體的屬性（特性），及語句間的邏輯關係等，或是修飾其它概念中介。上面的例子中，「A 實體.是.B 實體」的「是」就是一個概念中介，下面有更多例子：

本店.鳳梨.糖果.由.100.%.天然.鮮果.製成

[本店.鳳梨.糖果].由.[100.%.天然.鮮果].製成

塑膠.製品.使用.後.，.如果.未.妥善.回收.處理.，.將.對.環境.產生.衝擊.。

[塑膠.製品].使用.後.，.如果.未.妥善.回收.處理.，.將.對. [環境].產生.衝擊.。

[塑膠.製品].被.[人].使用.後

如果.[人].未.妥善.回收.處理.[塑膠.製品]

[塑膠.製品].將.對. [環境].產生.衝擊

第一句中，「由...製成」是一個概念中介，第二句中，「被...使用」、「...後」、「未...」、「妥善...」、「回收.處理...」、「將...」、「對...產生.衝擊」是描述實體間的關係或修飾其它概念中介。「如果.A.，.B」是描述 A、B 語句之間的邏輯關係。

這裡可以看到，概念中介的其中一個特點是，它不一定是語句中連續的一部份，像是「被...使用」、「對...產生.衝擊」這些概念中介在語句中都被其他實體或概念中介分隔開來，但它們仍是「單一一個概念中介」。

認識語句中實體及概念中介的定義，可以讓我們更清楚地檢視對語句進行點號標記時的標記方式是否合理，以及自下而上完成多次斜線標記時的運作邏輯。

實體在認知當中具有特殊概念，正確在語句中分離出實體是相當重要的；概念中介則是被設計來處理語句中除了實體以外其餘部分的一種認知工具。

3.2 知識樹

關於一個實體，可以有很多層面、不同方向上的認識。描述一個實體時，也會使用到非常多同義、不同義的真值語句。

關於橘子，我們可以說：

橘子.是.一種.水果

橘子.通常.是.橘色.的

橘子.有.皮

有些人.喜歡.吃.橘子

橘子.有.子

橘子.有.一種.特別.的.香味

這些真值語句描述了關於橘子的一些事實，而且是當我們要對「橘子」這樣的概念進行描述時，會自然而然產生的一些描述。

一方面，在建構一個智能系統時，我們希望能夠方便地檢視互有相關的真值語句，並且填補尚未得到的資訊，另一方面，我們也觀察到針對實體的常見屬性進行描述這樣的任務，對於多數人來說都是相當自然且迅速的。因此，我們可以使用知識樹的概念，來讓智能系統對於圍繞一個實體產生的相關認知更為明確。

知識樹的一種表達方式如下（多種表達方式都能達到類似效果）：

橘子（/水果）

是否有皮？　　有

是否有子？　　有

是否有香味？　　有

顏色　　通常是橘色

…

橘子的知識樹架構，是透過斜線表記由水果處繼承而來：

水果

是否有皮？　　？

是否有子？　　？

是否有香味？　　？

顏色　　？

…

我們不追求斜線表記可以一路向上追溯到一個共同根源，使得實體間的繼承關係可以明確地像大自然中的樹一樣，由一個共同的根開始開枝散葉。反之，我們只注重這種繼承關係是否對於描述對於實體的認知有所幫助。

當一個類別繼承了另一個類別後，屬性可能會增加或減少：

動物

生活在陸地上？　？

食性　　？

顏色　　？

體表有毛？　　？

…

而狗的類別在繼承了動物類別後，則有不同的屬性：

狗（/動物）

生活在陸地上？　是

食性　　雜食性

顏色　　？

體表有毛？　　不一定

品種　　？

可以牧羊　　？

…

從上方關於狗的知識樹節點，我們可以得到一些真值語句：

狗.生活.在.陸地.上

狗.是.雜食性.動物

狗.不一定.有.毛

另外請注意到，知識樹在高階智能系統理論當中是一個過渡性的設計，用來作為 LIT 架構中 T 系統的其中一種表達方式。事實上，多數「描述實體的真值語句的集合」，其功能都可以透過域的概念來實現，相關介紹請見「域」章節。

3.3 T 系統的功能

經由感官等資訊來源得到的真值，以及推論出來的真值，必定存放在某個地方。在 LIT 架構中，我們加上 T 系統來存放真值。

T 系統的功能類似於記憶，除了存放真值以外，T 系統也將彼此相關的真值整合起來，形成對於特定實體的瞭解，比如：

汽車：

有四個輪子

有方向盤

可以坐人

可以裝載貨物

由汽油驅動

上面的理解，有些精確，有些不精確。在電動車問世之前，汽車「由汽油驅動」或許是偏向精確的理解，但電動車普及後，一旦認識到電動車的存在，可能就必須將對於「汽車」「由汽油驅動」的理解修改為「由汽油或電力驅動」。

一個智能個體的 T 系統內容會隨著其經驗（比如來自感官的資訊輸入）與學習過程而不斷改變，初次認識到一類新的實體時，所存放的關於該實體的真值較

少，結構也較為粗糙。隨著與該實體的相關互動及認識增加，有關該類實體的真值也隨之增加，並且，透過這些真值語句所形成的結構也漸次演變地更加複雜且精細，並可能更貼近於該實體的真正特質。

－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－

比方說，一個智能個體可能會經歷這樣的認識過程：

由「拿鐵是一種咖啡」，得到

拿鐵：

　　是一種咖啡

由「紅茶拿鐵是一種拿鐵」，得到

紅茶拿鐵：

　　是一種拿鐵

並且經過 I 系統中的相關規則，比如：

「1/物.是.一種.2/物 -> （2/物.3/ST -> 1/物.3/ST）」

其中 ST 是代指任意類別的一個類別名稱。將「1/物」處代入「紅茶.拿鐵」，「2/物」處代入「拿鐵」，得到：

「紅茶.拿鐵.是.一種.拿鐵 -> （拿鐵.1/ST -> 紅茶.拿鐵.1/ST）」

這裡因為只剩下一個斜線，所以「1/ST」使用編號「1」，而不是「3」。因為已知「紅茶拿鐵是一種拿鐵」有真值，所以再得到：

「拿鐵.1/ST -> 紅茶.拿鐵.1/ST」

又因為「1/ST」處可以代入「是.一種.咖啡」，所以得到「紅茶.拿鐵.是.一種.咖啡」。

而個體的 I 系統中可能有這個真值：「我喝咖啡會不舒服」，且根據可用的相關規則，得到：「我喝紅茶拿鐵會不舒服」。

－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－

如果我們在此先假設

（1）「紅茶拿鐵」其實是「紅茶加上牛奶」

（2）該個體喝「紅茶」或「紅茶加上牛奶」都不會不舒服

那麼前述的「我喝紅茶拿鐵會不舒服」對於該個體而言就是一個錯誤的認識。

咖啡：

喝了會不舒服

紅茶拿鐵：

是一種拿鐵

是一種咖啡

喝了會不舒服

該個體後續認識到「紅茶拿鐵其實是紅茶加上牛奶」，而使「紅茶拿鐵是一種咖啡」不再具有真值後，「喝了會不舒服」的特性就從該個體對於紅茶拿鐵這類實體的認識中移除（同時，「紅茶拿鐵是一種拿鐵」、「紅茶拿鐵是一種咖啡」也一樣會被移除）。

經過這種實體認識的重塑後，對於該個體而言，紅茶拿鐵的認識可能變成：

鮮奶茶：

含有茶和牛奶

是我喜歡喝的飲料

紅茶拿鐵：

是一種鮮奶茶

含有茶和牛奶

是我喜歡喝的飲料

3.4 ECTSI 階層

ECTSI 階層是配合 L 系統的斜線標記需求，用以描述實體間關係的階層設計。

ECTSI 指的是這五個階層：

E（Entity，實體）

C（Category，集合）

T（Type，類別）

S（Sub-type，子類別）

I（Individual，個體）

必須先提及的是，這個階層設計具有任意的成分，定義並不嚴謹，且即使調整成更多階或更少階，也不會影響到它發揮作用。之所以設計了 ECTSI 階層，僅是為了方便「斜線標記」的使用者對於這種工具的應用方式有更加深入完整的理解。

在 ECTSI 階層設計當中，E 是最高層級，涵蓋的實體數量最多、範圍最廣、抽象程度最高；I 是最低層級，涵蓋的實體數量最少、抽象程度最小。

可以簡單理解成：「實體」、「存在」或「生物」等這類抽象程度非常高、涵蓋實體數量很多、其中個體差異很大的，更偏向 E 階層，「某個人」、「某支筆」、「某個段落」更偏向 I 階層。

一個實際的例子：

E（Entity，實體）：物/生物

C（Category，集合）：植物/花

T（Type，類別）：玫瑰花

S（Sub-type，子類別）：漂亮的紅色玫瑰花

I（Individual，個體）：窗邊的那朵漂亮的紅色玫瑰花

自下往上看，可以看到 I 階層是一個實體「個體」，T 階層是一個實體從屬的類別，介於其中的 S 是在個體以上、類別以下，包含了一群個體的概念。

從 T 階層再往上走，E 是「存在/物」這種涵蓋極廣的抽象概念，及距離 T 較

遠的高級概念，如「生物」、「無生物」、「元素」等，C 則介於 E 與 T 之間，是由具有共同點的一大類 T 所構成，但尚未到達 E 的寬廣程度。

再看到另一個例子：

E（Entity，實體）：物

C（Category，集合）：交通工具/車/電動車

T（Type，類別）：特斯拉電動車

S（Sub-type，子類別）：銀色特斯拉電動車

I（Individual，個體）：小明上週買的銀色特斯拉電動車

值得注意的是，ECTSI 階層中每一層究竟包含哪些概念，具有一定的模糊性。這一方面是因為它只是用以「協助斜線標記概念理解與操作」的指導框架，另一方面，也因為完善嚴謹的階層設計屬於智能系統的「最佳化」範疇，而一個既不完善、也不嚴謹的實體階層設計，仍然能夠支持智能系統的運作（讀者可以拿自身為例，看完善嚴謹的架構是否直覺性地存在腦海當中）。

因此，我們在此只對 ECTSI 階層做概念性的介紹，不再深入，讀者只需能夠利用 ECTSI 階層來對斜線標記自下而上的整段複數次標記過程產生更清晰的理解即可。

3.5 尖括弧與井字標記

我們可以使用尖括弧「<」與「>」在語句當中標示出實體：

小明.吃.了.兩.顆.橘子

<小明>.吃.了.兩.顆.<橘子>

當該實體是一個抽象集合（不是個體，而是一群個體或一類個體）時，可以將斜線標記置於尖括弧內：

小明.吃.了.兩.顆./橘子

<小明>.吃.了.兩.顆.</橘子>

表達繼承關係時，可以使用如下方法表示：

橘子.是.一種.水果

<小明>.吃.了.兩.顆.</橘子 ||i 水果>

其中，「||i」是「繼承自」某個集合的意思。

由於實體可能會有重名的現象，比如可能會有多個人都叫做小明，我們可以使用「#」給予 T 系統中的所有的實體編號，以使語句中實體的指稱對象明確：

<小明 #1>.吃.了.兩.顆.</橘子 ||i 水果>

看到另一個例子：

<小明.養.的.狗 ||i 狗 #2>.在.玩.球

我們在討論 LIT 架構的章節以外，也會繼續使用這些標記方式。

3.6 LIT 架構的功能與限制

我們已在第一章至第三章陸續介紹了 L 系統、I 系統，及 T 系統，這三個系統共同組成了 LIT 架構。

LIT 架構是一種系統設計，系統可以由多個子系統組成，而每個子系統也都是一個系統。採用 LIT 架構的系統，其內部就有 L 系統、I 系統，及 T 系統等子系統。

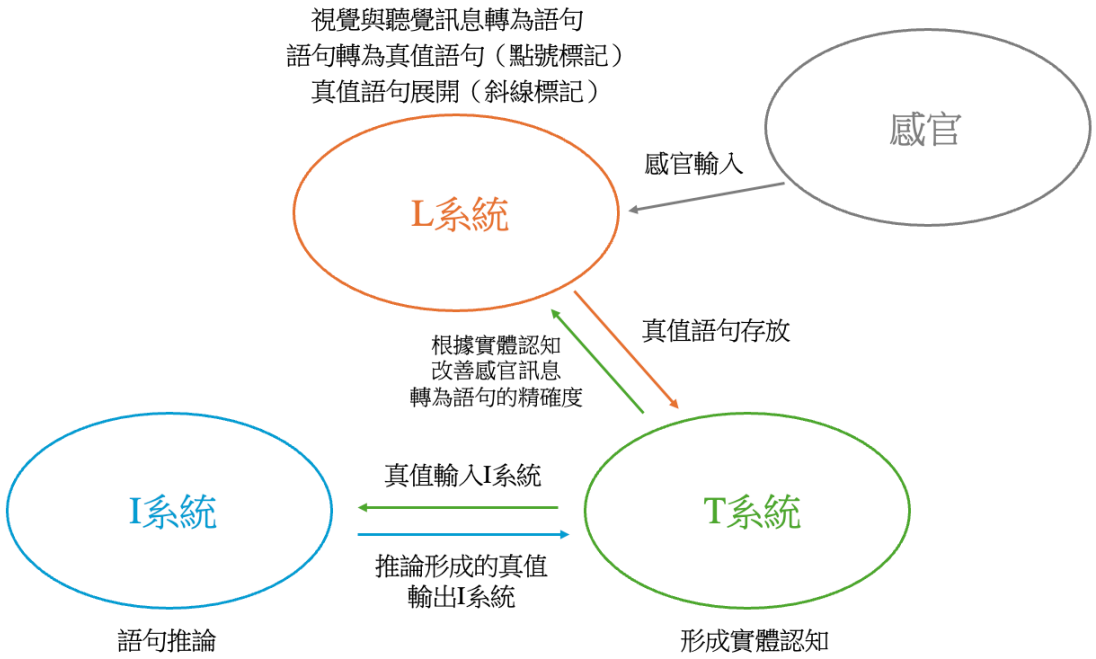
LIT 架構是 R.K.C 在研究初期時，基於對智能系統功能的觀察而設計出來的。T 系統的設計來自於對「記憶」功能與「實體認知」功能的觀察，L 系統來自於對「語言理解」功能的觀察，而 I 系統則來自對「推論」功能的觀察。這個架構標誌了高階智能系統理論發展初期，透過觀察智能系統功能、進行拆分與類比，而試圖實現具備這些常見功能的複合架構的嘗試。

LIT 架構的應用範圍有限，而且缺點非常明顯，比如無法清晰描述「經由學習產生的真值改變」、「某些真值只在有限的範圍內成立」等現象，同時，它也無法很好的描述系統的「預測」、「行動」、「錯誤」等行為表現。

不過，LIT 架構是發展出更完整的高階智能系統理論體系的重要基石，從觀察到 LIT 架構可解決與不能解決、可描述與不能完善描述之處出發，在後續發展出「域」與「系統」兩個概念後，R.K.C 得以將高階智能系統理論進一步推進與完善。

－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－

即便 LIT 架構的功用有限，我們仍然可以進一步討論 LIT 架構中三個主要子系統的功能與互動。



由上方的圖示可以看到，感官訊息先以影像或聲音方式，進入到 L 系統中，並且在轉為語句形式後，經過點號標記成為真值語句，再經過斜線標記，實現部分的真值語句展開（另一部份展開在 I 系統處發生）。

L 系統中產生的真值語句進入 T 系統後，結合成實體認知，並且透過 T 系統與 I 系統間的互動，將真值語句由 T 系統輸入 I 系統，進行推論後，得到的新真值語句再由 I 系統輸出，同樣存放在 T 系統。

值得注意的是，L 系統與 T 系統之間並非單向的關係，藉由 T 系統中存在的實體認知，並經由 I 系統的推論結果加強，在避免 T 系統中語句矛盾的情況下，可以改善 L 系統在將感官訊息轉為語句時的精確度。

比如，「我昨天吃了一顆很漂亮，顏色很（？）的蘋果」，如果「？」處聽不清楚，或者是筆跡模糊，仍然能夠推論出更可能是「紅」或「綠」，這是因為 T 系統中存放有相關真值的緣故。

同樣的，T 系統中存在的實體認知，也有助於 L 系統中發生點號標記及斜線標記的過程。

下雨天留客天留我不留

上面是一個著名的中文雙關語句，有下方幾個拆法（標記方式）：

下雨天，留客天，留我不留？

下雨，天留客，天留，我不留。

下雨天，留客天，留我不？留。

第一句是客人到了民屋後，向主人提問，第二句是主人拒絕，第三句是（儘管較為生硬）客人堅持希望主人接受留宿要求。

下雨.天.，.留.客.天.，.留.我.不.留.？

下雨.，.天.留.客.，.天.留.，.我.不.留.。

下雨.天.，.留.客.天.，.留.我.不.？.留.。

1/天氣_天.，. 2/動作_天.，.3/動作.4/人.不.3/動作.？

1/天氣.，.天.2/動作.，.天.3/動作.，.我.不.4/動作.。

1/天氣_天.，. 2/動作_天.，. 3/動作.4/人.不.？.5/動作.。

可以看到，三種解讀方式在經過斜線標記後的結果並不相同，這也就指出了，同樣一個無標記的輸入語句「下雨天留客天留我不留」，在當下對於這些實體的認識不同下，將會影響到 L 系統的解讀結果。

再看一個例子：

城市的夜晚點亮燈紅酒綠的世界

試著以兩個不同的方式標記此語句：

城市.的.夜晚.點亮.燈紅酒綠.的.世界（「城市」.「的」.「夜晚」.「點亮」.「燈紅酒綠」.「的」.「世界」）

城市.的.夜.晚點.亮.燈.紅酒.綠.的.世界（「城市」.「的」.「夜」.「晚點」.

「亮.燈」.「紅酒」.「綠」.「的」.「世界」）

如果今天是一個完全不懂此語言的人，手中僅有一本辭典，且辭典中僅列詞，不列詞的詞性及解釋，那麼他很可能無法分辨上述兩個標記方式何者正確。然而瞭解此語言的人，在認識「晚點」、「紅酒」等詞的意思之下，就能夠判斷第一個分法正確、第二個分法不正確（因為使用第二個標記方式時，在其認知中語句無法展開，也就無法「理解意思」）。

這讓我們看到 T 系統中對於實體的認識，可以協助 L 系統更正確地處理感官資訊中含有的語句輸入。

第四章：域

4.1 真值的效力邊界

有一些真值語句的真值顯然並不總是成立，比如：

- 今天.陽光.普照
- 街.對面.就是.一家.咖啡廳

如果我們把這類語句當作前提，並經由推論規則得到其他真值語句時，就同樣必須考慮真值是否成立的問題：

If 今天.陽光.普照 A

-> 今天.是.晴天 B

If 今天.是.晴天 B

-> 小明.要.帶. 陽傘 C

If 今天.不.是.晴天 D

-> 小明.不.帶.陽傘 E

當「今天.陽光.普照」成立時，我們會得到「小明.要.帶.陽傘」的結果，但是 A 並非總是成立，也並非總是不成立，我們需要找到一種方式讓 A 的真值效力只發揮在一定範圍以內，為了達到這個效果，我們引進「域」的概念。

域是一些真值語句的集合，這些語句在域當中都具有真值，但在域外則不一定。我們可以使用 {a}、{b}、…，即 {小寫英文字母} 來表示域，由於語句是以大寫英文字母來表示，在不會造成混淆時，我們有時也會單以小寫英文字母 a、b、… 來表示域。

{a}{A.=.今天.陽光.普照}

{a}{A}

上面這兩種寫法都代表 A 語句存在於 {a} 域當中。有時我們說「{a} 域」，有時說「a 域」、「{a}」，這些都表示同樣的概念。

在 {a}{A} 中，{a} 後方所接的 {A} 代表 A 語句在 a 域中，如果有 {a}{B, C}，則代表 B 和 C 都在 a 域中。

存在於域當中的所有真值，都是域的「元素」。

－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－

推論規則也可以存在域中，比如：

{a}{

If 今天.是.晴天 B

-> 小明.要.帶.洋傘 C

}

許多規則並不是在所有情況下都總是成立：如果小明今天要出門，或許確實需要帶洋傘，但是不出門的話就不需要，所以推論規則的效力也是有限的，只及於其所存在的域當中。也就是說，在包含這個規則的域中，推論規則就有效，在域外，推論規則不一定有效（有真值）。

如果域中不存在一個推論規則，那麼即使那個規則的前提都存在域內，也不能在這個域中透過這個規則來得到推論結果。類似地，如果推論規則在域中存在，但是作為其前提的某些真值語句不存在域當中，那麼同樣不能透過這個規則得到推論結果。

4.2 域的表記

前一節中，我們已提到使用如 {a}、{b}、…，即 {小寫英文字母} 來表示域。本節進一步介紹其它一些與域相關的表記方式。

- {a} 是域，也可以寫作 a 域（讀作小 a 域，以在口語上更明確地與

用在語句的 A 區隔）。形如 {A} 或 {B,C} 的也是域，前者是一個含有 A 語句的域，後者是同時含有 B 及 C 語句的域。

- 在 {B,C} 中，A 是否含有真值？答案是「不知道」。在 {A,B,C} 中，A 有真值；在 {B,C,～A} 中，A 沒有真值，但是在 {B,C} 中，A 的真值「未定」。
- {a}={B,C} 代表 a 是一個域，而且 a 中有 B 及 C 語句，而沒有其它既非 B 也非 C ，也不能由 B 與 C 推論出來的語句。{a}={B,C} 也可以寫作 {a}{B,C}。
- {a}={b} 代表 a 域與 b 域相等，即在 a 中的語句都存在 b 中，在 b 中的語句都存在 a 中，同時，不在 a 中的語句都不在 b 中，反之亦然。
- 當一個討論中同時出現 a 域及 b 域時，a 與 b 是否相等？一般來說，如果使用不同的小寫英文字母，我們預設這兩個域不相等，除非經過某些運算後顯示它們相等，或者討論者明示它們相等。
- {a} ≠ {b} 代表 a 不等於 b，即下列兩者並非同時成立：（1）所有在 a 中的語句都存在 b 中（2）所有在 b 中的語句都存在 a 中

－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－

- {a}{A,B} 代表 a 中含有 A 與 B，{C} 則是一個含有 C 語句的域。{a}+C 或 {a}+{C}，結果都是 {A,B,C}。要指代運算出的域時，可以另起一個名稱，如使用 {a'} 或 {b}，即運算結果為 {a'}{A,B,C} 或 {b}{A,B,C} 。如此一來，運算也可以這樣表達：{a}+C = {a'}{A,B,C}，類似的，也可以寫作 {a}+{C} = {a'}{A,B,C}、和 {a}+{C} = {b}{A,B,C} 等。
- {b}{A,B,C} 代表 b 中含有 A、B 與 C，{C} 是一個含有 C 語句的域。{b} – C 或 {b} – {C}，結果都是 {A,B}。
- {c}{A,B} 代表 c 中含有 A 與 B，{c} – C 或 {c} – {C}，結果都是 {A,B,-C}，或者寫成 {A,B,～C}。{c} – C 與 {c} – {C}，都等同於 {c}+(～C) 或 {c}+{～C}。
- 注意，{b}{A,B,C} – {C} 與 {c}{A,B} – {C} 的結果並不相同，{b'}={b}{A,B,C}–{C}={A,B} 中，C 沒有真值，～C 也沒有真值；

{c'}={c}{A,B}-{C}={A,B,-C}={A,B,～C} 中，C 沒有真值，～C 有真值。

－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－

- {a} = {A,B,C}，{b} = {…a,D}，代表 {b}={a}+{D}={A,B,C,D}；{c}={…a,-C,D} 代表 {c}={a}+{-C,D}={A,B,D}；{d}={…a,-D,E} 代表 {d}={a}+{-D,E}={A,B,C,～D,E}，我們將「…」稱為「展開符號」。
- {a}={A,B}，{b}={-B,C}，此時{a}+{b}為何？{a}+{b}={A,B}+{-B,C}={A,B,-B,C}={A,C}。{a}+{b} 中，B 無真值，～B 也無真值。
- 域可以存在於其它的域中，比如 {b} 可以存在 {a} 中。例如：{a}{{b}{A,B},C,D} 代表 {b}{A,B} 在 {a} 中，C、D 在 a 中，但不在 b 中。
- 由{a}{{b}{A,B},C,D}，我們可以進一步得到：

  {a}{b}{A,B}

  {a}={{b}{A,B}}+{C,D}
- {e}={{c}{A,B,～C}}+{C,D}={{c}{A,B,～C},C,D}。注意到，{e}{{c}{A,B,～C},C,D} 既不等於 {{c}{A,B},D}，也不等於 {{c}{A,B,～C},D} 或 {{c}{A,B},C,D}，因為已知 (～C) 在 {c} 中成立，但是在 e 中不一定成立（需視是否也在 c 中而定），所以這裡的 ～C 和 C 既不同時相消，也不能使用外部的 C 消掉內部的～C，或者內部的 ～C 消掉外部的 C。

域的表記方式是應操作需求而設計，讀者只需對這些表記方式有基本掌握，以便更佳瞭解域的概念及運作，並使後續章節的理解順暢即可。

4.3 兩種推論方式

另一種語句真值的侷限來自於狀態的改變。比如：

桌.上.有.三.顆.球　　C

如果「小明.從.桌.上.拿走.了.兩.顆.球（F）」，那麼桌上剩下一顆球，或者說：

桌.上.有.一.顆.球　　D

但是域中可能有一個顯而易見的規則

{a}　[R_1]

If　A’

->　～B’

A’　1/地方.有.2/數字.3/單位.4/物

B’　1/地方.有.5/數字.3/單位.4/物

使得在域 a 當中，C、D、R_1 不能同時成立。

有時候，如此例當中所示，我們會在推論矩陣當中的「->」下方加上數行，這幾行前方會寫出已出現於矩陣中的大寫字母，後方則寫出語句，這代表前方的大寫字母指的是後方的語句。這是為了在複雜的規則當中，避免矩陣的前提與結論這幾行過於擁擠。

－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－

為了描述域的狀態「變遷」，我們引入另一種推論方式，使得域的狀態能夠隨事件發生而變化。要紀錄原先的域發生改變時，本質上，我們是透過產生另一個新的域來描述改變後的狀態。

{a}{C, R_1}

{a}{C, R_1} [+F]>> {b}

{b}{D, R_1}

上面幾行表示的是在域 a 當中，有 C 語句與 R_1 推論規則，經過 F 事件，域 a 轉變為域 b，也可簡寫為 {a} >> {b}。

我們引入推論符號「>>」來表示這種新的推論方式，「經過 F 事件」寫為 [+F]，[+F]>> 代表域經過 F 事件發生的改變，也可以簡寫為 [F]>>。

{a} [R_2]

If 1/地方.有.2/數字.3/單位.4/物 C’

2/數字.-. 6/數字.=.7/數字 G’

\+ 5/人.從.1/地方.拿走.6/數字.3/單位.4/物 F’

>> {b} {…a, -C’, D’}

D’ 1/地方.有.7/數字.3/單位.4/物

從上面的推論規則看來，如果我們開始時有

{a}{C, R_1, R_2}（C=「桌.上.有.三.顆.球」）

並且在 {a} 中發生事件 F（F=「小明.從.桌.上.拿走.了.兩.顆.球」），那麼由規則 R_2，我們可以得到 {a} [+F] >> {b}，且有 {b}{D, R_1, R_2}（D=「桌.上.有.一.顆.球」）。

這裡，由於加上了「事件發生」及「狀態改變」的概念，在推論矩陣中 If 下方的第一列，我們加上了 + 來代表發生事件，並以「>>」取代原本的「->」來表示使用的是另一種（「>>」）推論方式。

這個例子中，域的變遷對應了日常生活中的表述：桌上原本有三顆球（C），小明拿走了兩顆球之後（[+F]），剩下一顆球（D）。

也就是說，一個智能系統認知到以上的敘述之後，會建立兩個域，包含 {a}{C} 與 {b}{D}，並且紀錄 {a} [+F]>> {b}。

4.4 超域與子域

超域與子域描述了域與域之間的包含關係。

{a}{A,B,C} 是 {b}{A,B} 的超域，{b} 是 {a} 的子域，因為每個在 b 中的元素都在 a 中。

{c}{A,B,～C} 也是 {b}{A,B} 的超域，但 {d}{B,～C} 不是 {b}{A,B} 的超域，因為 b 中的 A 不在 d 中。

{e}{{f}{A,B},C,D} 是 {{f}{A}} 的超域，也是 {{f}{A,B}}、{{f}{A},C} 的超域，但不是 {{f}{A,B,C}} 或 {{f}{A},B,C}} 的超域，因為 {{C}} 不在 e 中（在 e 中的是 C 或寫成 {C}，不是 {{C}}）、{B} 也不在 e 中（在 e 中的是 {{B}}，不是 B 或寫成 {B}），這種區別的用處在「域前綴」一節中可以更清楚地看出來。

另外，還有一個概念是「外部域」與「內部域」之間的關係。觀察 {e}{{f}{A,B},C,D} 與 {f}{A,B}（注意，這裡寫的不是 {{f}{A}}，沒有最外面的 {}），它們之間的關係是：e 是 f 的外部域，C、D 都在 f 的外部域中；f 是 e 的內部域，A、B 都在 e 的內部域中。

4.5 緻密域

由於域中的真值語句可以透過推論規則來在域中得到其他真值語句，因此域可能是處於一個「還未包含其中所有可推論出的真值」的狀態，也可能處於「已包含所有可推論出的真值」的狀態。

當域 a 中包含所有從其元素可推論出的真值時，我們稱 a 為緻密域，此時 a=a*（或記為 {a}={a}*，* 是在此處新引入的符號），a* 是 a 達到緻密域時的狀態，例如：

{a}{B, C, R_1}

[R_1] = B^C->E

{a}{B, C, E, R_1} = {a}*

以及

{b}{F, R_2, R_3, I}

[R_2] = F->J

[R_3] = I^J->K

{b}{F, R_2, R_3, I, J, K} = {b}*

緻密域 a* 含有域 a 中所有可推論出的真值語句，因此如果語句 A 可以透過

a 中的某些元素，經過有限次推論而得，那麼 A 必定也存在 a* 中。

4.6 同構域

同構域指的是兩個域當中的真值語句皆相同。

如果 {a} 與 {b} 為同構域，那麼在 a 當中具有真值的語句，在 b 當中皆具有真值，而且在 a 當中不具有真值的語句，在 b 當中都不具有真值（即 {a}={b}）。

由於推論規則也總是可以使用語句（真值）的方式來表述，因此，在 a 當中成立的規則，在 b 當中也都成立，在 a 當中不成立的，在 b 當中也不成立。

－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－

因為真值語句可以透過本身的展開來得到其他語句，因此需要特別注意 {a}* 與 {b}* 為同構域時，不一定也有 {a}={b}，比如：

{a}{A, B, R_1}

R_1 = A^B->C

{a}*={A, B, R_1, C}

{b}={A, B, R_1, C}={b}*

上面可以看出，雖然 C 在 b 中，也在 a 的緻密域 a* 中，但不在 a 中，使得 {a} ≠ {b}。

4.7 基底域

如果

（1）{b}*={a}*，且

（2）不存在一個 {c}，使得

（2-1）{b} ≠ {c}，又

（2-2）{c} 可在任意步推論後成為 {b}

那麼 {b} 是 {a} 的基底域。

可以理解為：{b} 是可以經過推論成為 {a} 的域當中，最根本的那個，因為 {c} 代指的對象包含所有可以經過推論後（先成為 {b} 後，再）成為 {a} 的域，但是我們找不到這樣一個比 {b} 更根本的 {c} 了，所以 {b} 就是最根本的那個。

－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－

比如：

{a}{A, B, R_1, C}

{b}{A, B, R_1}

R_1 = A^B->C

{b}={A, B, R_1, C}={a}*

且不存在

{c}≠{b}，又

{c} $[->]^n$ ={b}

n=1,2,3…

那麼 {b} 是 {a} 的基底域。

$[->]^n$ 的表記方式平常不會用到，在這裡，它只是代表 {c}「經過任意有限正整數次推論」後，與 {b} 同構（={b}）。

－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－

為了確認不存在這樣的 {c}，使得 {c} 可在任意步推論後成為 {b}，我們可以針對 {b} 中的規則使用「逆推論」，比如：

{b} {A, B, R_1}

[規則 R_1] = A^B -> D

[逆規則 R_1^R] = D <- A^B

這裡，R_1^R 指的是 R_1 的逆規則，利用 R_1^R 進行推論，相對於利用 R_1 進行推論而言，即是進行「逆推論」。由於 b 中不存在逆規則 R_1^R 的前提 D，且 b 中的逆規則只有 R_1^R，因此不存在域 c ，使得域 c 在若干次推論後等於 b。

而觀察：

{a} {A, B, R_2, D}

[規則 R_2] = A^B -> D

[逆規則 R_2^R] = D <- A^B

由於 a 中有 D 語句，滿足 R_2^R 的所有前提，所以我們知道必定有一個域 {A, B, R_2, D}+{-D}={A, B, R_2} 可以經過有限次推論（這裡是一次，即 A^B->D）後得到 a，也就是說，a 不是 a* 的基底域，但是 a 的基底域仍然會是 a* 的基底域。我們也可以使用 a^B 來表示 a 的基底域，且有 $a^B=(a^*)^B$。

一定要小心的是，從規則 R_2=A^B->D 導出的逆規則 R_2^R 是 D <- A^B，不是 D-> A^B！也就是說，我們只能得到：

{a}{A, B, R_2, D} [R_2^R]>> {b}{A, B, R_2}

而不能得到：

{c}{R_2, D} [R_2^R]>> {d}{A, B, R_2, D} //這是錯的

－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－

逆規則的構造方式，即是把原先的規則中所有的前提放到逆推論符號「<-」的後方，並把原先規則中所有的結論放到「<-」的前方，如 R_2^R 與 R_2 所示。

使用 R_2 與 R_2^R 進行推論時，可以得到如下的結果：

{c}={A,B,R_2,D}

{d}={A,B,R_2}

{d} [R_2]-> {c}

{c} [R_2^R]<- {d}

4.8 @符號與域前綴

@ 符號用於將域的「佈置因子」提出。當域中的真值都有共同的時間、地點、動作實體或描述對象時，我們就將這些因子稱為「佈置因子」，並且可以使用 @ 符號將他們提出到域的內容之前，代表域中的所有真值都在這些因子的佈置之下。

先看到這個例子：

{a}{

{a_1 || T_1, P_1}{A, B},

{a_2 || T_1, P_2}{C, D}

}

在 a 中，有 a_1 和 a_2 兩個域，且 a_1 的佈置因子為 T_1 及 P_1，a_2 的佈置因子為 T_1 及 P_2，代表 A 和 B 都在 T_1、P_1 的佈置下發生，C 和 D 都在 T_1、P_2 的佈置下發生。

我們可以將 a 寫成如下含前綴的表示方式：

{a}

@{T_1}{

{a_1 || P_1}{A, B},

{a_2 || P_2}{C, D}

}

因為 a 中所有的真值都有 T_1 前提，所以我們可以把 T_1 提出到域之前，成為 @{T_1} 前綴。上面使用 @{T_1} 前綴的寫法與此意義相同：

{a || T_1}{

{a_1 || P_1}{A, B},

{a_2 || P_2}{C, D}

}

使用 @ 符號還是使用 ‖ 符號，端看欲將前綴因子寫在 {域的名稱} 外，還是寫在其內而定，{a ‖ T_1} 和 {a}@{T_1} 的意義與效果沒有不同。

再看到以下的域：

{a'}

@{T_1}

@{P_1}{

{a_1}{A, B}

}

從前面的 {a} 中，我們試圖提取出子域 {a'}。由於 a' 的前綴當中包含 T_1 及 P_1，而在 {a} 中，只有 {a}{a_1} 中的 A, B 同時有 T_1 及 P_1 兩個佈置因子，{a}{a_2} 中的 C, D 則並沒有同時具有 T_1 及 P_1 兩個佈置因子（只有 T_1，沒有 P_1），所以 a' 中沒有包含 {a_2}{C, D}，這也使得 a ≠ a'。

類似的，可以試圖提出包含不同前綴的子域 a''：

{a''}

@{T_1}

@{P_2}{

{a_2}{C, D}

}

並且發現 {a} = {{a' ‖ T_1, P_1}}+{{a'' ‖ T_1, P_2}}，且 {a ‖ T_1} = {{a' ‖ P_1}}+{{a'' ‖ P_2}}。

－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－

{a}{

{a_1 ‖ T_1, P_1}{A, B},

{a_2 ‖ T_1, P_2}{C, D}

}

改為將 P_1、P_2 自 a 提出時，形成 a 的子域時，寫成如下的表示：

{a'''}

@{P_1}

{

{a_1 || T_1}{A, B}

}

{a'^4}

@{P_2}

{

{a_2 || T_1}{C, D}

}

可以看出 {a} = {a''' || P_1}+{a'^4 || P_2}。a''' 和 a'^4 都是 a 的子域，a 則是 a''' 和 a'^4 的超域。

－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－

使用一個實例來說明：

{a}{

{a_1 || T_1, P_1}{A, B},

{a_2 || T_1, P_2}{C, D}

}

{a}{

A=小明昨天早上在學校上課

B=小明昨天早上很開心

C=小美昨天早上在宿舍整理行李

D=小美昨天早上很興奮

}

{a}{

{a_1 ‖ 昨天早上, 在小明的學校}{

A=小明上課

B=小明很開心

},

{a_2 ‖ 昨天早上, 在小美的宿舍}{

A=小美整理行李

B=小美很興奮

}

}

{a’}

@{昨天早上}

@{在小明的學校}{

{a_1}{

A=小明上課

B=小明很開心

}

}

{a”}

@{昨天早上}

@{在小美的宿舍}{

{a_2}{

A=小美整理行李

B=小美很興奮

}

}

這裡可以發現：

{a} = {{a'‖ 昨天早上, 在小明的學校}} + {{a"‖ 昨天早上, 在小美的宿舍}}

{a‖ 昨天早上} = {{a'‖ 在小明的學校}} + {{a"‖ 在小美的宿舍}}

與前面抽象的例子中，觀察到的 {a} = {{a' ‖ T_1, P_1}}+{{a" ‖ T_1, P_2}} 與 {a ‖ T_1} = {{a' ‖ P_1}}+{{a" ‖ P_2}} 相同。

4.9 演序與演序合併

演序指的是域在差時推論「>>」下的一連串變化，無論經時長短、變化幅度，從開始一連串變化的「起始域」至變化結束時的「停止域」，及變化過程當中的每個「快照」，都包含在演序當中。

演序的概念是為了讓我們能更著眼於域的動態變化，而非僅關注其靜態的特性。在智能系統當中，對於演序的認識，是預測外界變化及建立行動策略的基礎。

先看到下方這一連串域的變化：

{a ‖ T_1}{A, B, C}

{a} [+E_1]>> {b ‖ T_2}{A, B, D}

{b} [+E_2]>> {c ‖ T_3}{A, B, C}

上面的例子中，{a}>>{b}>>{c}，或者寫為 a>>b>>c 就是一個演序。

以實例說明（為了專注在演序的概念上，不使用點號標記）：

{a ‖ T_1}{

桌上有一個紅色按鈕，桌上有一個黃色按鈕，螢幕上顯示了一顆蘋果

}

{a} [+按下桌上的黃色按鈕]>> {b ‖ T_2}{

桌上有一個紅色按鈕，桌上有一個黃色按鈕，螢幕上顯示了一顆柳橙

}

{b} [+按下桌上的紅色按鈕]>> {c ‖ T_3}{

桌上有一個紅色按鈕，桌上有一個黃色按鈕，螢幕上顯示了一顆蘋果

}

注意到，{a} 和 {c} 域的內容（元素集合）相同，但其前綴 {‖ T_1} 與 {‖ T_3} 不同，因此 {a ‖ T_1}={c ‖ T_3}，但是 {a}≠{c}。

－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－

演序合併指的是將演序經由「串聯」操作合併成新的演序，或經由「並聯」操作合併成演序集合。

先看到串聯的例子：

[演序 1][

a >> b >> c

]

[演序 2][

b >> c >> d >> e

]

我們可以將演序 1 與演序 2 串聯成：

[演序 3][

a >> b >> c >> d >> e

]

並且寫為 [演序 3] = [演序 1]+[演序 2]，當「+」前後皆為演序時，代表對這個操作子前後的演序進行串聯。

像上方的例子一樣進行串聯時，必須確定兩個演序有重疊的部分（上例中，是 b>>c 的部分），且一個演序的起始域在重疊的部分前端（b>>c>>d>>e 的起始域 b 在 b>>c 的前端），另一個演序的停止域在重疊的部分後端（a>>b>>c 的停止域 c 在 b>>c 的後端）。同時，重疊的部分在兩個域中的每個快照（上例中，就是 b 和 c）的緻密域都必須與另一個域中對應的快照的緻密域同構。

－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－

再看到並聯的例子：

[演序 4][

f >> g >> h

]

[演序 5][

f >> g >> i >> j

]

我們可以將演序 4 與演序 5 並聯成演序集合：

[演序集合 6][

[演序 4][f >> g >> h]

[演序 5][f >> g >> i >> j]

]

上面的並聯結果可以寫為 [演序集合 6] = [演序 1]（||）[演序 2]，其中「(||)」代表域的「並聯」。域並聯時，兩域的起始域必須相同。

由兩個域的演序並聯而成的演序集合中，會有重疊的部分與不重疊的部分，重疊的部分也會有不相同處。比如上面例子中，如果 h 的緻密域與 i 的緻密域不同構，那麼演序的分岔就開始於 h 與 i，造成演序分岔處的域不同構的因子，叫做「差異因子」，請見下方的例子。

如果我們已經找到差異因子（比如可能是佈置因子 T_n 或 P_n）來解釋為何有時觀察到「f >> g >> h」演序，有時則觀察到「f >> g >> i」演序，而且 h 的緻密域與 i 的緻密域不同構，那我們就可以寫出如下的表示：

[演序集合 6][

[演序 4 || $diff_1$][f >> g >> h]

[演序 5 || $diff_2$][f >> g >> i >> j]

]

其中，$diff_1$ 和 $diff_2$ 是演序 4 和演序 5 在演序集合 6 中的「差異域」，差異域代表雖然不在各自演序的域快照當中，但會使得演序變化過程不同的真值（同樣地，可能是佈置因子 T_n 或 P_n）的集合。這些真值不在 f、g 等域當中，但在他們的外部域中，即存在：

{ …$diff_1$, f>>g>>h}

也存在

{ …$diff_2$, f>>g>>i>>j}

其中「…」是「域的表記方式」一節中提到過的「域的展開符號」。

假設我們知道差異因子是 A 與 ～A，比如：

A.=.<角色 #1>.的.魔力.值.小於.100

～A.=.～(<角色 #1>.的.魔力.值.小於.100)

[演序集合 6][

[演序 4 || {A}][

{f}{B.=.<角色#1>.在.怪物.旁} [<我>.按下.對戰.鍵]>>

{g}{B, C.=.<角色#1>.面對.怪物} [<我>.按下.攻擊.鍵]>>

{h}{B, C, ～D.=.～(<角色#1>.施放.技能)}

]

[演序 5 ‖ {～A}][

{f}{B.=.<角色#1>.在.怪物.旁} [<我>.按下.對戰.鍵]>>

{g}{B, C.=.<角色#1>.面對.怪物} [<我>.按下.攻擊.鍵]>>

{i}{B, C, D.=.<角色#1>.正在.施放.技能} >>

{j}{B, C, E.=.怪物.血量.減少}

]

]

上面這個例子當中，{ …diff$_1$, f>>g>>h} 和 { …diff2, f>>g>>i>>j} 指的是：

{

A, {f}{B.=.<角色#1>.在.怪物.旁}

} [<我>.按下.對戰.鍵]>> {

A, {g}{B, C.=.<角色#1>.面對.怪物}

} [<我>.按下.攻擊.鍵]>> {

A, {h}{B, C, ～D.=.～(<角色#1>.施放.技能)}

}

和

{

～A, {f}{B.=.<角色#1>.在.怪物.旁}

} [<我>.按下.對戰.鍵]>> {

～A, {g}{B, C.=.<角色#1>.面對.怪物}

} [<我>.按下.攻擊.鍵]>> {

～A, {i}{B, C, D.=.<角色#1>.正在.施放.技能}

}　>> {

～A, {j}{B, C, E.=.怪物.血量.減少}

}

－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－

找到差異因子來解釋為何「並聯而成的演序集合」當中，各自演序的變化過程不同後，系統就可以依據「當前域中含有的差異因子為何」來預測域的變化。

當無法推論出差異因子來解釋變化過程的不同時，系統只能先將後續的不同變化可能性看作是機率性發生（依照經驗當中各種情況的發生頻率/次數），或者試圖採取行動以找到差異因子。相關的討論，請見「行動」與「學習」章節。

第五章：系統

5.1 系統與表示域

在高階智能系統理論的範疇內，所謂「系統」，是指自外部域接受真值，並且對外部域產生影響的真值集合。

討論系統時，我們著眼於這幾個方向：

- 系統自外部接受的真值與接受的方式
- 系統的內部變化
- 系統對外部域產生影響

將這些方向展開來看本書中一些相關的主題：

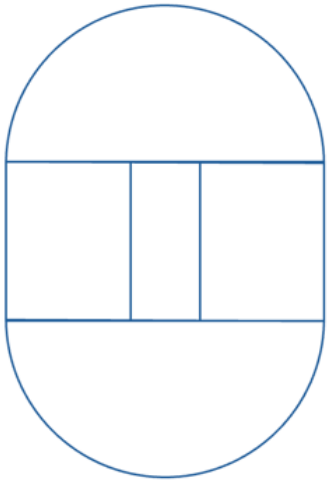
系統自外部接受的真值與接受的方式	• 感官 • L 系統 • 演序 • 觀察域 • 對話 • 注意力域
系統的內部變化	• I 系統 • 學習 • 定形與定式 • 範式 • T 系統 • 記憶與遺忘 • 推論與預測 • 情感
系統對外部域產生影響	• 策略 • 行動 • 功能 • 對話 • 情感

我們可以發現，本書中的重要主題，都圍繞在系統的幾個重要的討論方向上，其中如對話等主題，則橫跨了數個討論方向。上方的列表是為了幫助讀者先產生高層次的理解，而非具確定性的嚴謹分類。

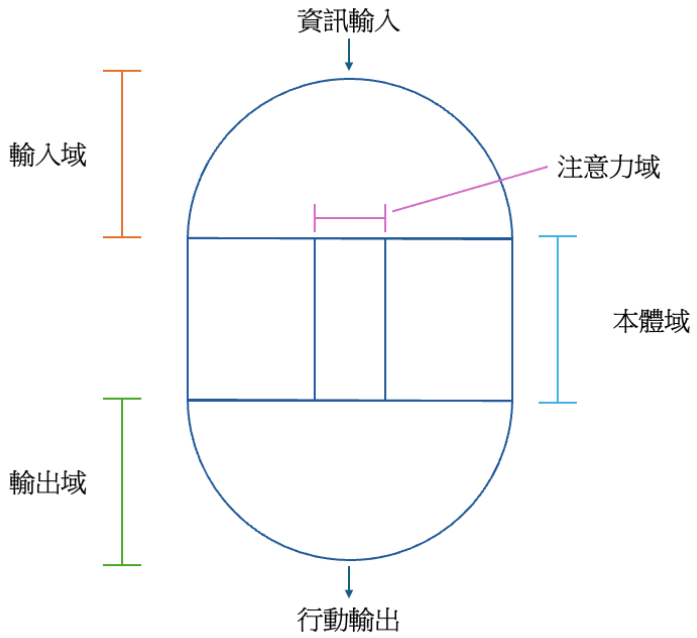
－－

系統是一些真值的集合，因此，我們也可以使用域的方式來描述一個系統。系統的「表示域」是「內容為系統中的真值集合的域」，已經提過的、適用於域的規則及特性，都適用於系統的表示域。

我們使用一個簡易的圖形來描述系統的結構：



下方的圖示進一步指出系統結構的各個部分：



從圖示中可以看到，系統的輸入是「資訊」，輸出是「行動」，輸入的資訊經由輸入域來到本體域中，輸出的行動則由本體域經由輸出域輸出。

本體域中，有一個特別的域稱作「注意力域」，這是系統當下專注於處理的部份，我們會在「不完全系統的議題」一章中進行更多關於注意力域的討論。

輸入域、輸出域、本體域及注意力域都在系統的表示域當中，可以將其簡單理解為：

{系統的表示域}{

{輸入域}

{本體域}{

{注意力域}

}

{輸出域}

}

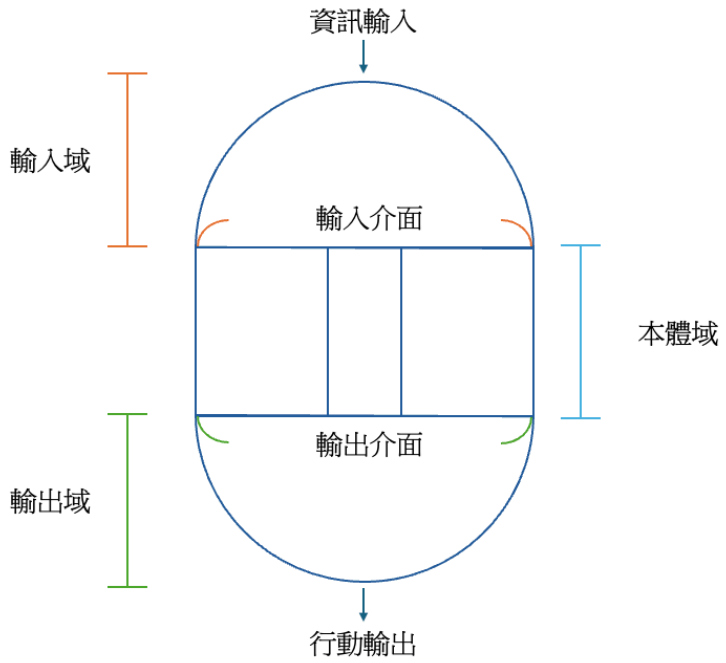
需要注意的是，表示域中的各個部分並非隔離的，比如說，真值會從輸入域中移出到輸入域與本體域的外部，再移入到本體域的內部，也會在本體域中，自注意力域的外部移入注意力域或自注意力域內部移出。

5.2 介面

進入系統輸入域的是由感官或其它輸入管道輸入的資訊，但這些資訊尚未經過解讀，是原始的訊息形式，如光、聲波、壓力、化學分子啟動的電訊號，或電子儀器接收到的電壓、像素等。

未經解讀的資訊不會完全進入到系統的本體域中，而會經過系統的「輸入介面」，成為本體域可解讀的真值類型，之後才會進一步在本體域中與其它的真值產生交互。在經過輸入介面時，許多輸入域中的真值會被濾除，使得進入本體域的資訊量遠遠小於透過輸入管道進入到輸入域的資訊量。

輸入介面也是一個「域」，其中的真值（規則也是一類真值）描述了輸入域中的真值如何從其原本的形式轉換成本體域中的真值形式。輸入域中的資訊不一定都適合使用自然語言描述，我們使用「輸入譜」來描述進入輸入介面的訊息。相關內容請見「高階理論的實踐方法」一章中有關「感官」的討論。



一個系統若不輸出資訊，那麼我們不僅無法以其功能推測出結構，對它進行建構與分析的意義也甚微，這是在高階智能系統理論當中。將系統定義為狹義的「自外部域接受真值，並且對外部域產生影響的真值集合」的原因。

系統輸出資訊的過程，我們稱之為「行動」。在本體域中，有一類規則（Do 規則，相關討論見該節）會決定系統將要對外輸出的資訊內容，這些資訊內容會成為輸出介面的輸入，也就是說，這些資訊內容會自本體域進入到輸出介面。

輸出介面也是一個域，其中的真值都與「本體域輸出的資訊」轉化為「影響外部域的真值」相關，在進入輸出介面時，這些資訊還是適合以自然語言表達的形式，自輸出介面向輸出域移動時，這些真值已經被轉化為並不適合以自然語言表達的形式。

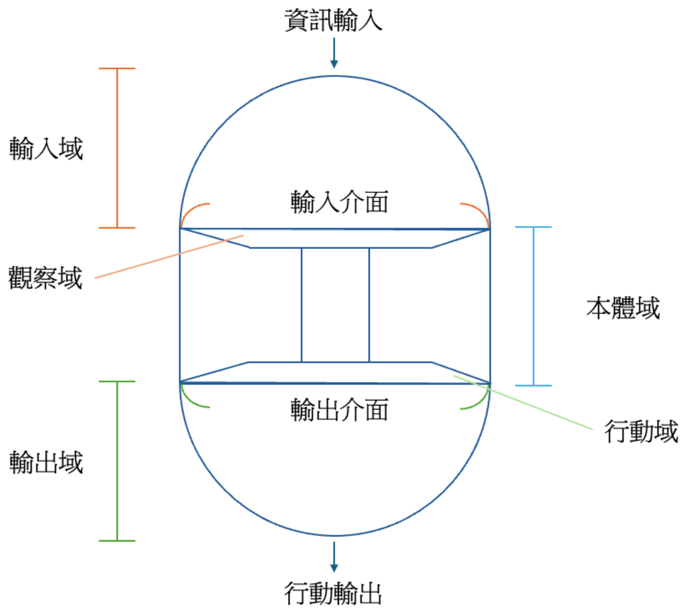
我們使用「輸出譜」來描述自輸出介面輸出的訊息，這些訊息在輸出域中，進一步轉化為控制系統可直接影響的物質（身體或位元等）的訊號，比如電訊號、化學訊號等形式，再表現為系統可被外部觀察到的「行動」。這些行動會影響到外部域中的真值，也會進一步影響到外部域中的其它系統。

有關輸入到「輸出介面」中的資訊、自輸出介面輸出的「輸出譜」，與行動輸出的方式，請見「行動」與「高階理論的實踐方法」等章節中的討論。

5.3 觀察域與行動域

在智能系統當中，觀察域與行動域是「本體域」當中的域：觀察域自輸入介面輸入資訊，並輸出資訊至本體域；行動域則自本體域輸入資訊，且輸出資訊至系統的輸出介面。

由於觀察域、行動域都是既有輸入、也有輸出的域，因此也可以把它們看作是「觀察系統」與「行動系統」，「觀察系統」與「行動系統」各自的表示域，就是觀察域與行動域。不是所有的系統都有觀察域與行動域，但它們通常是「智能系統」構造的一部份。



在高階智能系統理論中，我們以「自然語言形式」來描述進入觀察域的資訊與觀察域輸出的資訊，以便於使用高階方法來探討其中的真值及它們的變化；同樣的，我們也以自然語言形式來描述進入行動域的資訊與行動域輸出的資訊。

－－－－－－－－－－－－－－－－－－－－－－－－－－－－

資訊進入系統時，會經過多次的「解讀」，解讀指的是將真值轉換為另一種形式，以成為一些規則的前提並與其它真值互動。

輸入介面會過濾掉的資訊中，包含了系統即使在資源充足時，也無法解讀的資訊（這不是指感官無法接收的資訊，這些在進入輸入域的過程中就已經濾除），以及系統因資源不足而未解讀（可以解讀，但沒有解讀）的資訊。

觀察域則過濾掉因系統的資源不足，雖然在輸入介面的輸出當中，但並未進入

到本體域中其它部分的真值。簡單地理解，就是如「雖然聽到了對方說話，當下也有聽懂，但直接忘記」，或者「雖然看到了場景中的一些物體，並且知道那些物體是什麼，但是並沒有花心思去注意它們」等的這些情形。

在系統的架構中設計了觀察域，是為了解釋「輸入介面的輸出真值集合」（所有從輸入管道進入的訊號，轉換為適於使用自然語言表示的真值形式後的結果）與「實際會與本體域中其它部分互動的真值集合」之間的差異。

有關系統面臨到的資源限制，將在「不完全系統的議題」章節中深入討論。

5.4 真值穿透與 Do 規則

{a}{

A, B, C, D,

{b || T_1}{R_{AC1}=A'^C'->E'}

{c || T_2}{R_{BD1}=B'^D'->F'}

}

{A}=@{T_1}{A_1}

{B}=@{T_2}{B_1}

{C}=@{T_1}{C_1}

{D}=@{T_1}{D_1}

為了介紹真值穿透的概念，先看到上方的例子。a 域中有 A、B、C、D，這裡，這些語句使用了大寫的英文字母來標記，但是它們本質上也是「未帶前綴的域」。舉個例子來說：A 可能是「昨天早上小明在學校上課」，A_1 可能是「小明在學校上課」，而 T_1 是「昨天早上」。

因為沒有另行設計「語句的前綴」的概念，而只設計了「域的前綴」，所以這裡我們將 A 以只含有它自己一個元素的域 {A} 來表示，並且有 {A}=@{T_1}{A'}。在 a 當中，因為 {A} 沒有前綴，所以也可以直接寫作 A。

另外，a 中還有內部域 b 與 c，其中 b 有前綴 T_1、c 有前綴 T_2，b 中的推論規則 R_{AC1} 是一個與 A 和 C 有關的抽象規則，R_{BD1} 是一個與 B 和 D 有關的抽象規則。

{D}=@{T_1}{D_1} 中，前綴是 T_1 而非 T_2，這是因應下方例子所需，不是筆誤。

先看到 A_1 與 B_1 如何穿透到 b 與 c 域當中：

{a}{

C,

D,

{b || T_1}{A_1, R_{AC1}=A'^C'->E'}

{c || T_2}{B_1, R_{BD1}=B'^D'->F'}

}

因為在 a 中有 A，即帶前綴 T_1 的 A_1，而內部域 b 也帶前綴 T_1，所以在「前綴對齊」的條件滿足的情況下，A_1 可以穿透到 b 當中。類似地，B_1 也可以穿透到 c 當中。

我們也可以用和域（或稱並域）的方式來表達真值的穿透：

{a}{

@{T_1}{C_1}={m || T_1}{C_1},

@{T_1}{D_1}={n || T_1}{D_1},

{b || T_1}{A_1, R_{AC1}=A'^C'->E'}

{c || T_2}{B_1, R_{BD1}=B'^D'->F'}

}

{a}{

{n || T_1}{D_1},

{p || T_1}={m || T_1}+{b || T_1}={A_1, C_1, R_{AC1}=A'^C'->E'}

{c ‖ T_2}{B_1, R_{BD1}=B'^D'->F'}

}

由於滿足「真值對齊」的條件，{m ‖ T_1}（m 是我們在這裡幫 @{T_1}{C_1} 新起的名字）可以和 {b ‖ T_1} 取和，得到新的域 {p ‖ T_1}（p 是這個新的域的名字）。要使用「穿透」的概念，還是「和域」的概念，取決於操作者本身的偏好而定，其實質相同。

－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－

有時候，系統中的一些域有隱藏的前綴，這些前綴存在域上，但是我們為了方便起見沒有將它們寫出。這時，需要外部域中有相應的穿透規則，才能夠使真值穿透進出，而不能只是看到前綴似乎簡單對上，或者是空白的，就進行穿透操作。後續章節的討論中，時常會遇到這類情形。

因此，一般來說，如果外部域的真值沒有相同前綴，我們不預設其能穿透到域中。如下的兩個域：

{a_1}{

A,

{b_1 ‖ @}{B, C}

}

{a_2}{

A,

{b_2 ‖ @}{A, B, C}

}

因為 {a_2} 中的 {b_2} 與 {a_1} 中的 {b_1} 不相等，且我們又不預設在 a_1 當中，A 能從 {b_1} 外穿透到 {b_1} 中，因此，我們會預設 {a_1} 與 {a_2} 不相等，除非其他證據顯示它們相等。

這裡，{b_1 ‖ @} 和 {b_2 ‖ @} 中的 @ 指的是 b_1 與 b_2 域的所有前綴，雖然同樣使用 @ 符號，但與「域前綴」一節中的使用方式不相同，是另一個獨立的符號用法。

{c_1}{

D,

{d_1}{E, F}

}

{c_2}{

D,

{d_2}{D, E, F}

}

雖然 {d_1} 與 {d_2} 的前綴並未被明確寫出，但我們並不預設域的前綴是「真值的空集合」，因此 {d_1} = {d_1 || @}，{d_2} = {d_2 || @}，這兩個式子當中，@ 都可能是空集合，也可能不是空集合。

使用簡寫 {d_1} 及 {d_2} 時，由於我們不預設它們的前綴 @ 是空集合，也就使得 D 不能直接穿透進入 {d_1} 中，因此，我們預設上方列出的 {c_1} 與 {c_2} 不相等。

總而言之，透過上面介紹的真值穿透過程，真值可以從系統的行動域外部、本體域內部穿透到行動域內部，成為 Do 規則的前提，也可以在其它域的外部及內部之間移動。

－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－

有一類規則與行動相關，我們將它們稱作「Do 規則」，Do 規則與其它的規則差別在於它們的「推論結果」直接與系統的行動相關。

比如：

[Do 規則 D_1]

If	外面.下雨	A
	<我>.要.出門	B
Do	<我>.帶.傘.出門	C

[Do 規則 D_2]

If	<我>.肚子.餓	D
Do	<我>.找.食物.吃	E

可以看到，在其它規則使用「->」或「>>」符號的位置，在 Do 規則中使用的是「Do」。與「Do」符號同一行，或者在其下方行的真值，都是 Do 規則的推論結果。

Do 規則存在於行動域中，這些規則會從本體域當中的其它部分接收真值作為前提，並經過 Do 推論後產生 Do 真值。Do 真值並非全部都會再進入到輸出介面中，而是會經過行動域內部的變化後，只留下部分真值，並且每個時刻，又只將這些留下的真值的一部份輸出，成為輸出介面的輸入。有關「行動」產生的更多深入討論，請見「行動」章節。

5.5 功能

系統的功能是系統在變化時顯現的特性。

觀察系統的功能對於「構造」系統（這裡，構造是一個動作）有很大的幫助，因為當兩個系統「功能同構」時，其中一個系統具備的功能，另一個系統也會具備。有關系統的同構，請見「同構系統」一節。

一般的智能系統通常具有（不限於）這些功能：

- 記憶
- 學習
- 推論與預測
- 根據預測選擇行動
- 解讀自外界輸入的資訊

由於在構造系統時，我們設定為目標的智能系統通常都具有這些功能，所以為了使構造出來的智能系統與目標系統「功能同構」，就必須考慮如何在構造時使系統同樣具有這些特性。

「根據特定的輸入（給定的資訊及任務），給出特定的輸出（執行行為）」也是

一類功能，這類功能是由上面不同類型的功能複合而成，可能使用到解讀資訊、記憶、推論與預測、根據預測選擇行動等功能。

－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－

功能之間，有「功能支配」的關係。

當系統具有一個功能 A，代表該系統必須具有功能 B 時，我們就稱功能 A「支配」了功能 B。換句話說，如果功能 A「支配」了功能 B，那麼系統必須要先具有功能 B，才能具有功能 A。

舉例而言，看到以下的複合功能。

功能 A：

「列出 1 至 100 間含 1、100 的範圍中所有的質數，再列出同範圍所有的合數」

功能 B：

「列出 1 至 100 間含 1、100 的範圍中所有的質數」

功能 C：

「列出 1 至 100 間含 1、100 的範圍中所有的合數」

我們可以很容易地看出功能 A 支配功能 B 與功能 C，因為若一個系統不具有功能 B，或不具有功能 C ，它就不具有功能 A。

再看到一個例子。

功能 D：

「將一副撲克牌中共 13 張花色為方塊的牌按點數大排到小」

功能 E：

「指出一副撲克牌中共 13 張花色為方塊的牌中點數最小者」

功能 D 支配功能 E，因為若一個系統具有功能 D，那麼在完成功能 D 時，可以同時附帶完成功能 E。我們也可以指出「系統必須要具有功能 E，才能具有功能 D」，因為並不存在系統具有功能 D，卻不具有功能 E 的這種情形（畢竟，具有功能 D 時，就會實現功能 E）。

－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－

探討功能之間的支配關係，有助於簡化建構「功能同構」系統的過程。比如，要建立同時具有上面例子中功能 D 及功能 E 的系統，只需先建立具有功能 D 的系統；而要建立同時具有上面例子中功能 B 及功能 C 的系統，也只需先建立具有功能 A 的系統。

另外，若一個功能是幾個子功能間的同時或差時排列，如功能 A 為功能 B 與功能 C 的差時排列，那麼只需要我們構造出一個僅具有功能 B 的系統，並構造出一個僅具有功能 C 的系統，就可以透過系統的合成來構造出具有功能 A 的系統。有關系統的合成與構造，請見「構建系統與合成系統」、「構造系統與構造競爭」兩節。

5.6 同構系統

系統的「同構」指的是系統之間在某些特性上具有相同之處。

我們定義出下列幾個不同程度的同構，形成系統的「同構階層」：

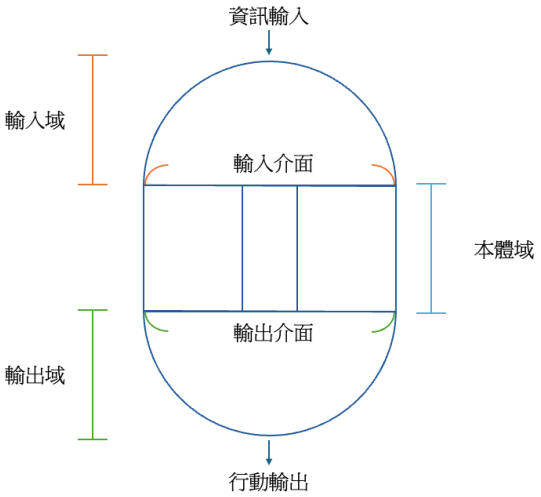
- 範圍同構
- 功能同構
- 表示同構
- 演序同構

－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－

在「介面」一節當中，我們提到過：「進入系統輸入域的是由感官或其它輸入管道輸入的資訊，但這些資訊尚未經過解讀，是原始的訊息形式，如光、聲波、壓力、化學分子啟動的電訊號，或電子儀器接收到的電壓、像素等。

未經解讀的資訊不會完全進入到系統的本體域中，而會經過系統的「輸入介面」，成為其本體域可解讀的真值類型，之後才會進一步在本體域中與其它的真值產生交互。在經過輸入介面時，許多輸入域中的真值會被濾除，使得進入本體域的資訊量遠遠小於透過輸入管道進入輸入域的資訊量。」

其中，以本體域可解讀的真值類型呈現，且使得本體域狀態發生改變的值的集合，共同構成本體域的「輸入範圍」。



「範圍同構」的兩個系統之間，經由輸入介面進入系統本體域的輸入範圍，以及從本體域輸出，會再經由輸出介面進入輸入域的「行動範圍」皆相同。但是，範圍同構的系統在接收到完全相同的輸入時，輸出不一定相同。

「功能同構」的兩個系統之間必定範圍同構，而且，功能同構的系統在接收到（類型與值）完全相同的輸入時，輸出一定相同。但是，功能同構的兩個系統其表示域的緻密域不一定同構（可能同構，也可能不同構）。

「表示同構」的兩個系統之間必定功能同構，而且，表示同構的系統之間，表示域的緻密域也是同構的。但是，表示同構的系統 S(a) 和 S(b)，雖然它們的表示域 a 與 b 的緻密域之間，有 {a}*={b}*，但是不一定有 {a}={b}。

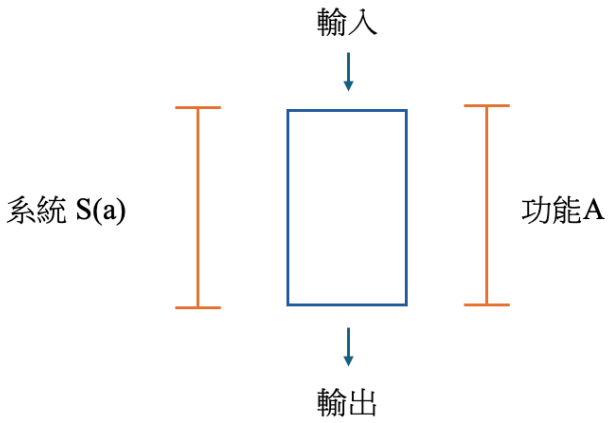
「演序同構」的兩個系統之間必定表示同構，而且，演序同構的系統 S(a) 和 S(b)，它們的表示域 a 與 b 之間，有 {a}={b}。演序同構的系統在接收到一連串完全相同的輸入時，不僅輸出相同，而且在接收輸入到輸出行動的過程當中，系統表示域改變而形成的演序完全相同。

比如，若系統 S(a) 和 S(b) 演序同構，S(a) 在接收到輸入「經過 F 事件」（[+F]）時的演序是 {a}>>{a_1}>>{a_2}，其中 {a_1} 是接收輸入後 S(a) 的表示域、{a_2} 是輸出行動時 S(a) 的表示域；系統 S(b) 接收到同樣的事件 [+F] 時的演序是 {b}>>{b_1}>>{b_2}，{b_1} 是接收輸入後 S(b) 的表示域、{b_2} 是輸出行動時 S(b) 的表示域，那麼，由於兩個系統演序同構，我們可以得到 {a}={b}、{a_1}={b_1}、{a_2}={b_2}。

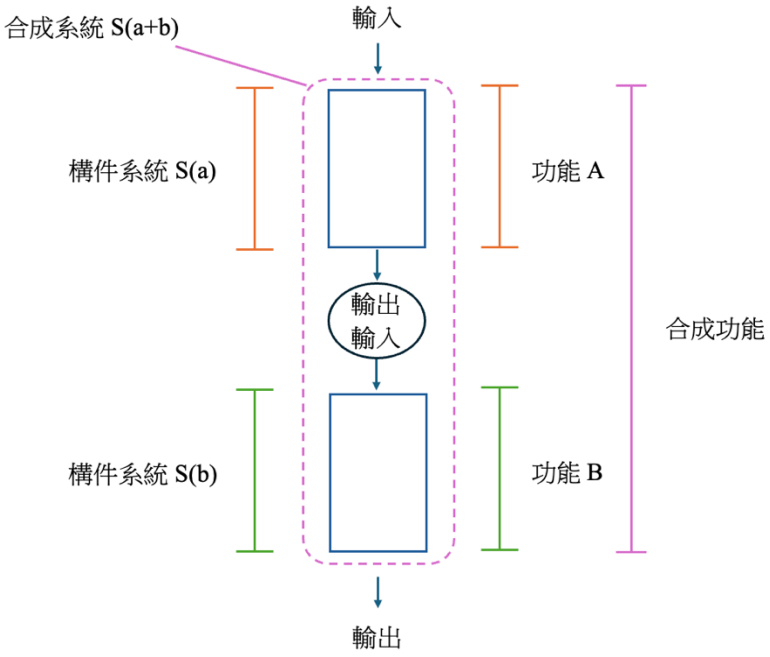
5.7 構件系統與合成系統

當一個系統 S(a) 是由複數個系統 $S(a_1)$、$S(a_2)$、$S(a_3)$、… 合成而成，我們就稱 $S(a_1)$、$S(a_2)$、$S(a_3)$、… 等系統為系統 S(a) 的「構件系統」，而 S(a) 為這些構件的「合成系統」。

本節中，會使用如上所示的簡化系統圖例，顯示系統接收輸入並向外輸出。

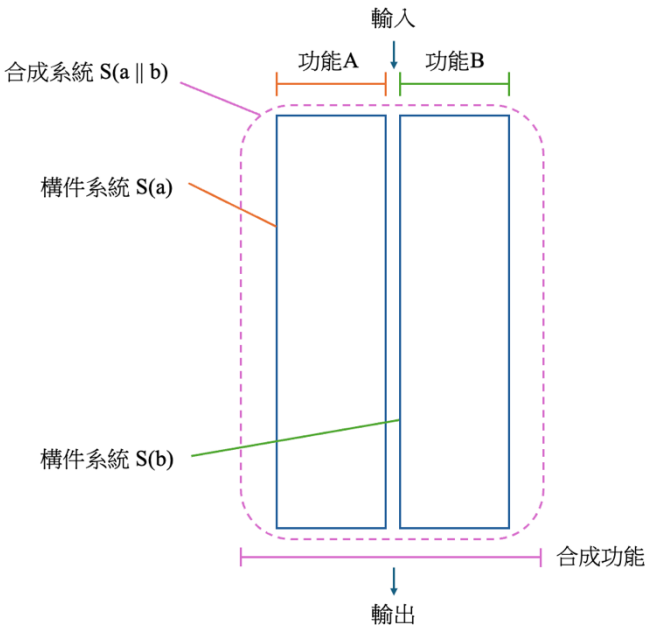


上方這個圖例中，我們可以看到系統 S(a) 接收輸入並向外輸出，而表現出功能 A。



合成系統時，我們可將構件系統進行串聯與並聯。上方的圖例是將構件系統 S(a) 與 S(b) 進行「串聯」，形成合成系統 S(a+b)，這裡「+」是串聯的意思。

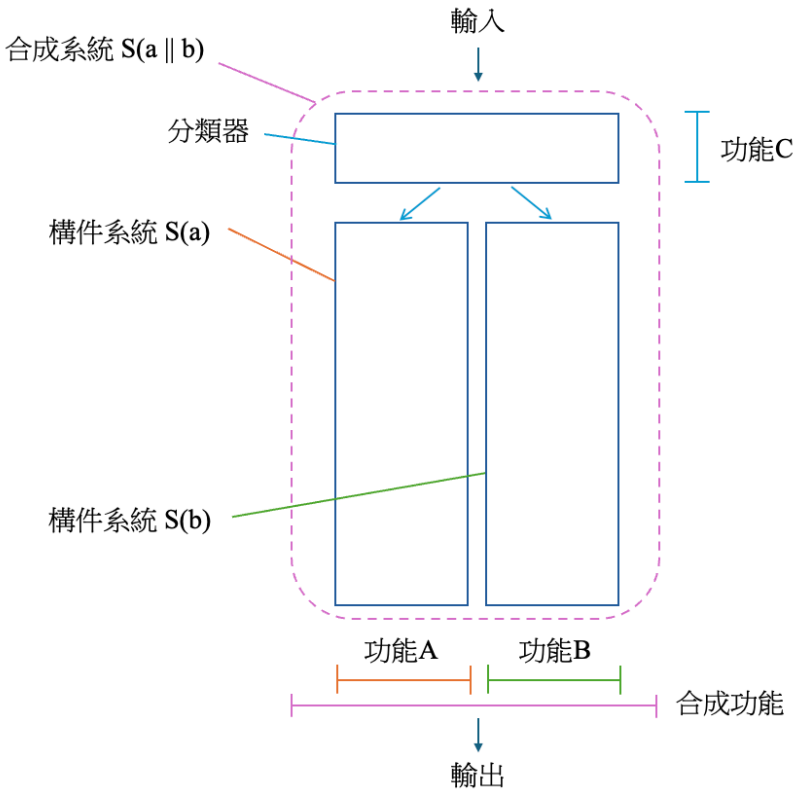
可以發現到，當我們以圖中的順序串聯 S(A) 與 S(B) 時，S(A) 的輸出成為 S(B) 的輸入，整個合成系統 S(A+B) 表現出的功能是功能 A 與功能 B 的串聯合成。



上方的圖例則是將構件系統 S(A) 與 S(B) 進行「並聯」，形成合成系統 S(A || B)，這裡「||」是並聯的意思。

可以發現到，當我們以圖中的方式並聯 S(A) 與 S(B) 時，S(A) 的輸入與 S(B) 的輸入都是系統輸入的一部份，整個合成系統 S(A || B) 的輸出是由 S(A) 與 S(B) 的輸出共同組合而成。

然而，合成系統 S(A || B) 的輸入究竟是進入構件 S(A) 還是 S(B)，仍然需要一個方法來決定，因此在「並聯合成」系統當中，我們需要在被並聯的構件前方再加上一個「分類器」。由於分類器也是一個系統，因此這樣做相當於將「分類器構件」與「構件 S(A) 及 S(B) 的並聯系統」串聯起來。



從上方的圖例中可以看到，合成系統的輸入成為分類器的輸入，分類器的功能就是將合成系統的輸入分類為構件 S(A) 的輸入或 S(B) 的輸入，因此分類器的輸出是「將輸入指派給系統 S(A) 或系統 S(B)」的「行動」。

5.8 系統的有效度與可解釋度

評估系統在特定功能上的表現時，可以使用許多不同的指標。其中，我們可以將系統針對特定功能的有效度與可解釋度結合起來，成為兩段式的評估方法。

在評估系統在一個功能上的表現時，我們是將候選系統（現在正在進行構造的系統）與目標系統（按照理想方式運作的系統）進行相互比較。

系統在特定功能上的有效度，介於 1 與 0 之間（或者 100% 與 0% 之間）。針對目標系統在這個特定功能上的表現，如果候選系統的表現與其完全相同，即給定同樣的輸入，輸出完全相同，那麼候選系統在此功能上的有效度為 1（100%），如果候選系統的表現與其完全不同，那麼候選系統在此功能上的有效度為 0（0%）。需注意的是，只有在範圍同構時，才能使用以上的方法進行有效度的評估。

當兩個候選系統對於同一個目標系統而言，對於特定功能的有效度不相上下時，我們就會進一步評估候選系統間何者有更高的可解釋度。我們將系統的可解釋度定義為：給定候選系統 $S(a)$，與在評估中的特定功能上與 $S(a)$ 有效度相同的系統 $S(k_1)$、$S(k_2)$、$S(k_3)$、…，所有這樣的 $S(k_n)$ 中可解釋度最高的 $S(k_H)$。當 $S(k_H)$ 的表示域已知，或者我們可以完全預測 $S(k_H)$ 在其所有功能的所有輸入範圍下的輸出時，就定義 $S(k_H)$ 的可解釋度為 100%。

－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－

從上面的定義中可以看出，目標系統的可解釋度，就是「與目標系統在該功能上功能同構的候選系統」當中，可解釋度最高者 $S(k_H)$ 本身的可解釋度。

也就是說，當我們能構造出一個「表示域已知」的系統 $S(k_H)$，它與某個系統 $S(a)$，在已知的所有功能上表現都完全相同，即 $S(k_H)$ 與 $S(a)$ 功能同構時，因為 $S(k_H)$ 的可解釋度為 100%，我們就認為 $S(a)$ 也具有 100% 的可解釋度，即使我們可能仍然無法構造出與 $S(a)$ 表示同構或演序同構的系統；相對的，可解釋度為 0 的意思是完全無法構造出一個「表示域已知」的系統 $S(k_H)$，使其功能與目標系統相同。

合成系統當中，$S(a+b)$ 在合成功能 A+B（A 是 $S(a)$ 的功能、B 是 $S(b)$ 的功能）上的可解釋度是 $S(a)$ 和 $S(b)$ 的可解釋度中較低者，$S(a \parallel b)$ 在功能 A 上的可解釋度是 $S(a)$ 在功能 A 上的可解釋度，在功能 B 上的可解釋度是 $S(b)$ 在功能 B 上的可解釋度（假設 $S(a)$ 具有功能 A、$S(b)$ 具有功能 B）。

另外，我們可以指定一個特定的計算方式，來決定同一個同一個功能的不同輸入範圍由不同的構件系統完成時，構件系統之間在這個功能上各自的可解釋度如何加權計算出合成系統在同一個功能上的可解釋度。比如，如果一個功能是「將輸入的整數乘以二、加一後輸出」，且我們定義其輸入的範圍是由 1 到 10 的 10 個整數，而根據系統中分類器的設計，當輸入為 1 到 4 時，由構件系統 S(a) 進行處理，當輸入為 5 到 10 時，由構件系統 S(b) 進行處理。

如果我們指定的計算方式是針對構件系統各自負責處理的輸入整數的種類總數進行加權平均，且 S(a) 的可解釋度為 1（100%），S(b) 的可解釋度為 0（0%），那麼系統 S(a || b) 在這個功能上的解釋度就是 [(100%*4) + (0%*6)]/10 = 0.4（40%）。針對不同的功能，我們可以指定不同的，適合當前討論功能的「特定計算方式」。

對於一般的智能系統而言，我們無從得知其表示域與內部演序，只能觀察其行為並推測其輸入，以判斷它具有的功能。當我們能夠針對一個智能系統構造出一個表示域已知的系統，且其在所有我們感興趣的功能上都與這個智能系統表現相同，我們就認為這個智能系統已可解釋。

5.9 構造系統與構造競爭

對系統進行分析的目的是對系統的行為進行預測。對系統的行為進行預測的方法，即是觀察系統的功能，並透過構造出功能相同的系統，來判斷在其它特定的輸入下這個系統的輸出。

構造系統時，我們可以依循以下的步驟：

- 列出目標系統的功能清單
- 使用「已知表示域的系統」覆蓋功能清單的一部份
- 使用「已知表示域的系統間的合成」覆蓋尚未覆蓋的功能清單的一部份
- 使用「未知表示域的系統當中，可解釋度最高者」覆蓋仍未覆蓋的功能
- 找到所使用的所有構件系統間的合成方式，並構造分類器
- 測試合成系統在目標功能清單上的有效度

－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－

如果我們要構造有功能 I、功能 II 及功能 III 的系統，來模擬一個同樣具有這三個功能的目標系統，且我們有以下可用的候選構件：

	在功能 I 上的 有效度/可解釋度	在功能 II 上的 有效度/可解釋度	在功能 III 上的 有效度/可解釋度
構件系統 S(a) （簡稱構件 A）	0.8/1	-	1/0.8
構件系統 S(b)	1/0.6	-	-
構件系統 S(c)	1/0.5	-	-
構件系統 S(d)	-	-	-
構件系統 S(e)	-	-	-
構件系統 S(f)	-	-	1/0.3

	在功能 II-I 上的 有效度/可解釋度	在功能 II-II 上的 有效度/可解釋度
構件系統 S(d) （簡稱構件 D）	1/0.8	-
構件系統 S(e)	-	1/0.7

其中，功能 II 是功能 II-I 與功能 II-II 的串聯合成。

- 列出目標系統的功能清單

我們已經有目標系統的功能清單，即「功能 I、功能 II 及功能 III」。

- 使用「已知表示域的系統」覆蓋功能清單的一部份

「覆蓋功能清單的一部份」意思是這個構件必須具有功能清單中的一些功能。所謂「具有」功能，即在該功能上的有效度高於一定的門檻（可以視需要設定）。

由候選構件中觀察，在功能 I 上具有一定有效度的是構件 A、構件 B 和構件 C，其中因為構件 B 在功能 I 上優於構件 C（有效度不低於構件 C、可解釋度不低於構件 C，且有效度或可解釋度至少有一者高於 C），因此我們不考慮採用構件 C。

接著比較構件 A 和構件 B，如果我們較重視構件 B 在功能 I 上的有效度是

1，「完整具有」這個功能，而構件 A 的有效度則不是 1，那麼我們可以優先考慮採用構件 B；反之，如果我們較重視構件 A 具有的「可解釋度較高」的特性的話，則可以優先考慮採用構件 A。在這裡，為了示範目的，我們先採用構件 B。

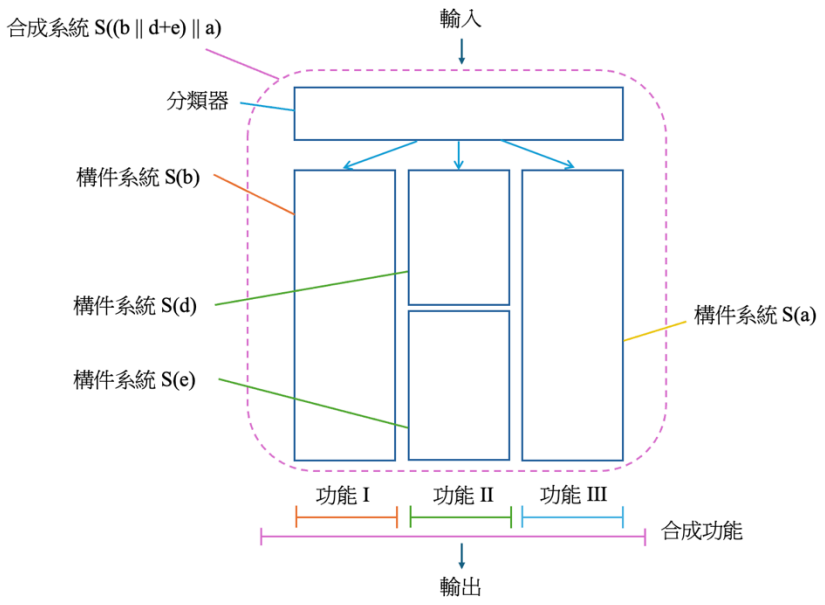
- 使用「已知表示域的系統間的合成」覆蓋尚未覆蓋的功能清單的一部份

接著，我們考慮功能 II。與功能 II 有關的是構件 D 及構件 E，且從候選構件中觀察，找不到單獨具有功能 II 的構件。由於功能 II 是功能 II-I 與功能 II-II 的串聯合成，我們可以使用構件 D 及構件 E 的串聯合成來覆蓋功能 II。構件 D 及構件 E 串聯合成的系統中，因為構件 D 的可解釋度是 0.8，構件 E 的可解釋度是 0.7，串聯而成的合成系統 S(D+E) 的可解釋度是兩者中較低者，即 0.7。

- 使用「未知表示域的系統當中，可解釋度最高者」覆蓋仍未覆蓋的功能

接著考慮功能 III。與功能 III 有關的是構件 A 及構件 F，由於構件 A 在功能 III 上優於構件 F，我們優先考慮使用功能 A。此時，因為構件 A 在功能 I 上也有非 0 的有效度，必須考慮分類器在功能 I 上的設計，以決定與功能 I 有關的輸入應該由構件 A 還是構件 B 處理。

由於我們可能希望由構件 B 而非構件 A 來處理功能 I 相關的輸入，此時，在合成系統 S((b ‖ d+e) ‖ a) 中，就必須設計分類器，將與功能 I 相關的輸入指派給構件 B。合成系統 S((b ‖ d+e) ‖ a) 的圖示如下：



－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－

同時，我們也可以觀察到，要構造具有功能 I、功能 II 及功能 III 的系統時，並非只有合成系統 S((b || d+e) || a) 這一種構造方法。

雖然構件 A 和構件 B 相比，在功能 I 上的有效度較低，但同時其在功能 I 上的可解釋度較高。另外，如果不使用構件 B，而只使用構件 S(d+e) 與 S(a)，那麼使用到的構件更少。

這代表，我們並非一定會選擇使用 S((b || d+e) || a)，而也可能會選擇使用 S(d+e || a)，且當我們認為難以提升構件 B 的可解釋度時尤其如此，反之，如果構件 B 的可解釋度易於提升，而構件 A 在維持可解釋度的前提下有效度難以提升，就更可能選擇優先使用 S((b || d+e) || a)。

選擇改善構件 B 在特定功能上的可解釋度時，就意味著我們必須以 S(b) 為目標系統，建構出一個同樣具有這些功能的合成系統 $S(b_2)$。

當構造出的合成系統 $S(b_2)$ 在功能 I 上的表現與 S(b) 完全一致，而可解釋度高於 S(b) 當前在功能 I 上的可解釋度時，S(b) 在功能 I 上的可解釋度也就提高至 $S(b_2)$ 在同功能上的可解釋度。同時，我們也可使用 $S(b_2)$ 來替換 S((b || d+e) || a) 中的構件 B，即轉為使用合成系統 $S((b_2 || d+e) || a)$。

如上面所討論的，當針對一個目標系統可以採用複數種構造方法，而這些構造方法之間互有優劣時，不同構造方法之間就形成了「構造競爭」。

第二部分 —— 不完全系統

第六章：不完全系統的議題

6.1 不完全系統的特性

完全系統指的是具有這些特性的系統：

（1）瞬時推論

（2）無限記憶

（3）推論無錯誤

而並非同時具備上述所有特性的系統，就是「不完全系統」。

－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－

在第一章至第五章當中，我們討論「系統」時，都假設系統具備完全系統的「完全性」，也就是上面列出的所有完全系統應具備的特性。然而，完全系統只存在於理論當中，現實層面上，我們所認識到的所有智能系統都是不完全系統，這是因為「瞬時推論」特性是難以達成的，至於「無限記憶」特性，雖然資源充沛的情況下，記憶空間限制常可以較為寬鬆，也仍然無法達到理論上的無限記憶空間。

「瞬時推論」指的是系統的表示域中，所有可進行的靜態推論（「->」）在瞬間完成，其推論出的結論真值可以再作為其它推論的前提而進一步進行推論時，下一步的推論也在瞬間完成，因此，瞬時推論的特性使得完全系統的表示域在任何時刻都是緻密域；動態推論（「>>」）由於是在事件發生時才使域的狀態發生改變，因此動態推論發生前，表示域為緻密域，事件發生且域中發生動態推論後，表示域在一個瞬間裡（時長趨近於零）是原表示域 $\{a_1\}$ 經過 [+E] 後的結果，即 $\{a_2\}=\{a_1\}[+E]$（也可寫成我們更熟悉的 $\{a_1\}$ [+E]>> $\{a_2\}$），$\{a_2\}$ 可能不是緻密域，但是下一個瞬間（與前一個瞬間可視為同一個瞬間，因前一段時長趨近於零），又完成 $\{a_2\}$ 當中可發生的所有靜態推論，達到 $\{a_2\}^*$。

「無限記憶」指的是系統的表示域容量無限。容量並非無限的系統，其表示域無法同時存放無限數量的真值，一般而言，現實世界中的智能系統都不是具備無限記憶特性的系統。我們將在「有限記憶」一節中進一步討論系統表示域中真值佔用的容量。

「推論無錯誤」指的是系統的表示域可完全按照其內真值進行正確推論，需注意的是，推論無錯誤指的並不是「真值」無錯誤。系統的表示域中仍可能存在與其所處的環境不相容的真值，這些真值對於該環境來說，也是「錯誤」的，但「推論無錯誤」特性指的是系統並不發生此類的錯誤（以靜態推論為例）：

$\{a_1\}\{R_A=A->B, A\}$

$\{a_1\}$ $[R_A]->$ $\{a_2\}\{R_A, A, C\}$

且 $\{\ldots(\{a_1\}+\{B\})\}^* \neq \{a_2\}^*$。

在上面的例子中，a 經過 R_A 規則的推論，應該要成為 $\{R_A, A, B\}$，但是在推論後，卻成為 $\{R_A, A, C\}$，而且 $\{R_A, A, B\}^* \neq \{R_A, A, C\}^*$（上面的 $\{\ldots(\{a_1\}+\{B\})\}$ 就是 $\{R_A, A, B\}$）。我們也將這類錯誤稱為「推論錯誤」。更多關於錯誤類型的討論，包括真值錯誤與推論錯誤等，請見「錯誤」章節。

－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－

不完全系統主要受制於以下幾個層面的限制：

（1）能量

（2）單步推論需時

（3）推論帶寬

（4）記憶容量

（5）絕對錯誤（，即推論錯誤）

系統具有的能量影響到其單位時間內可進行的推論步數。能量不夠時，單位時間內，單一推論鏈上進行的推論步數無法到達「時長/單步推論需時」。比如，如果單步推論需時為定值 0.1 秒，那麼能量充足時，單一推論鏈每秒可執行 10 步（為了簡化分析，假設每個推論步驟需時相同），但能量不充足時，則可能只能達到每秒執行 5 步，這 5 步可能是平均分散在 1 秒內完成，也可能較為集中在 1 秒當中的某一段或某幾段時間內完成。

推論帶寬代表系統可同時進行的推論鏈數量，或也可理解為「線程」。當系統的推論帶寬為 3 時，而每個推論鏈所需帶寬為 1，系統可能可以同時進行 3 個推論鏈上的推論（為了簡化分析，假設每個推論步驟所需帶寬相同）。當然，無論

帶寬充分或不充分，甚至無論能量充分或不充分，在同時進行多個推論鏈上的推論時，各個推論鏈都有可能無法保持僅執行該單一推論鏈時的推論速率。不過，能量不充分時，系統的推論速率肯定無法達到飽和推論速率，也就是說：

「單位時長內的所有推論鏈最高推論步數的加總」

無法達到

（系統總推論帶寬/平均推論鏈帶寬）x（單位時長/單步推論需時）
=（同時執行的推論鏈數量）x（單位時長內每推論鏈推論步數）」

記憶容量的限制前面已經有提及，而絕對錯誤（推論錯誤），指的就是該系統因其條件與特性而在推論時發生的錯誤，它與跟系統所處的外在環境有關的「相對錯誤」不同，是在推論「過程」發生，我們在「錯誤」一章中會進行討論。

－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－

我們已經看到，在一個不完全系統當中，單位時間內可執行的推論步數有限。也就是說，一個現實中的智能系統，時常必須只針對「感官輸入中的資訊」及「表示域中尚未完成的推論過程」進行選擇性的推論，而不能不經選擇全部進行。

注意力域、推論距離、路徑依賴等關鍵理論部分的發展，與不完全系統的概念產生至為相關。值得一提的是，如第一章中所述，「不完全系統」的概念定義，是在「注意力域」與「路徑依賴」、「範式」等本章介紹的重要概念發展完備後，才被清晰地描述出來。

6.2 抽象、實例與要件

在本節之前談論的「域」，有些屬於抽象域，有些則屬於實例域。系統觀察外界時解讀出的真值，在沒有使用斜線標記進行「抽象化」的情況下聚合在一起，就會形成「實例域」，比如：

$\{a_1\}$

{

小明.養.的.狗.是.柴犬,

小明.養.的.狗.的.體型.偏.中小.型,

小明.養.的.狗.的.耳朵.是.三角.形,

小明.養.的.狗.有.捲曲.的.尾巴

}

針對實例域當中某些可進行斜線標記的部分進行抽象化後，域就成為了「抽象域」。這種抽象域的產生，通常是系統將複數個彼此相似的實例域，透過抽象過程進行合併而來。

先看到一個與 a_1 相似的實例域：

{a_2}

{

小美.養.的.狗.是.柴犬

小美.養.的.狗.的.體型.偏.中小.型

小美.養.的.狗.的.耳朵.是.三角.形,

小美.養.的.狗.尾巴.是.直.的

}

如果系統觀察到 a_1 與 a_2，且因為某些觀察上的原因，在「是.柴犬」、「體型.偏.中小型」、「耳朵.是.三角.形」等真值上有一致的觀察，但「尾巴.是.捲曲.的」和「尾巴.是.直.的」上則沒有一致的觀察，那麼系統可以針對其中的「重複因子」中出現的實體進行抽象化，且在抽象化的過程中不納入不一致的「脫落因子」。對 a_1 與 a_2 進行抽象化的結果可能如下：

{a'}

{

1/狗.是.柴犬

1/狗.的.體型.偏.中小.型

1/狗.的.耳朵.是.三角.形,

}

系統當中含有這樣的抽象域時，我們會預期這個系統應該要在後續又觀察到

「柴犬」、「體型.偏.中小.型.的.狗」，或者「耳朵.是.三角.形.的.狗」時關注到這個域，並且「透過某隻狗是柴犬來推論出其他真值」，或者「透過某隻狗體型偏中小型來推論出他可能是柴犬」等。為了實現這種行為，我們可以考慮在抽象域內部加上一個「要件域」，裡面放入我們覺得合適的要件，另外，也可以把要件域中的某些元素加到前綴當中，比如，把「1/狗.是.柴犬」，或者把「1/狗.的.體型.偏.中小.型」放到前綴當中，以顯示出域中的真值是處於這個因子的佈置之下。

不過，系統當中很可能會有其他的域，標示了其他種類的狗（比如「科基」）同樣具有中小型的體型。這時，如果我們把「1/狗.的.體型.偏.中小.型」提到前綴當中時，就會有域是既有這個前綴，又有「1/狗.是.柴犬」在域中，而另外同時有域也有這個前綴，但是有「1/狗.是.科基」在域中；而若是轉而把「1/狗.是.柴犬」提到前綴當中，則符合這個前綴的域可能只有一個，且有「1/狗.的.體型.偏.中小.型」、「1/狗.的.耳朵.是.三角.形」等真值，可以做出確定性的推論。因此，我們選擇將「1/狗.是.柴犬」作為 a’ 的要件時，可以比將「1/狗.的.體型.偏.中小.型」作為要件時更確定地推論出更多真值。

－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－

下面是另一個實例域與抽象域的例子：

$\{b_1\}$

{

長方形.ABCD.的.長.是.6.公分,

長方形.ABCD.的.寬.是.3.公分,

6.公分.乘以.3.公分.是.18.平方.公分,

長方形.ABCD.的.面積.是.18.平方.公分,

}

透過學習與觀察，系統當中可能有許多類似 b_1 的實體域，它們之間，只有其中的數字部分不同。這時候，系統就可能將其中一些域抽象化，且把實體域中的同一實體以同一標號的斜線進行標示，形成如下方這樣的抽象域：

{b’}

{

<1/長方形>.的.長.是.2/數字.公分,

<1/長方形>.的.寬.是.3/數字.公分,

2/數字.公分.乘以.3/數字.公分.是.4/數字.平方.公分,

<1/長方形>.的.面積.是.4/數字.平方.公分,

}

形成了這樣的抽象域後，系統會將其他符合條件的觀察也代入到這樣的域當中，看看這些真值是否時常共同成立，以確定抽象域的適用範疇。是否在「1/長方形」前後加上尖括弧都可以，加上尖括弧時，可以將它連繫到系統對於實體的認知上。

根據存在假設，上面的域中，必須也存在「/長方形.的.長」、「/長方形.的.寬」與「/長方形.的.面積」等。因此，在選取要件時，我們可以自由選擇將「/長方形.的.面積」、「/長方形.的.長」、「/長方形.的.寬」中的一些放到要件域當中時，使系統可以有效率地進行推論。

－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－

含有「使用到斜線符號標記的真值」的域，就是一個抽象域。將抽象域中所有「使用到斜線符號標記的真值」以合法的方式代入詞組，使其不再使用到斜線符號標記時，形成的域就是該抽象域的一個「實例」。

抽象域中使用同一個斜線前數字的部分，在將域還原為實例域時需要代入同一個實體。

為了表示抽象域與其實例之間的關係，我們引入「%」符號來表示抽象：

b ∈ %a

「%a」是 a 這個實例域的抽象，因為真值可以經過多次斜線標記，所以 %a 有多種可能性，是一個域的集合。「∈」是「屬於」的意思，「b ∈ %a」代表 b 屬於「%a」這個域的集合，也就是說，b 是 a 對應的抽象域當中的一種。

({b} = {A’}) ∈ %{A}

「b = {A}」指的是 b 這個域中的元素有且只有 A’。按照慣例，使用到斜線標記的真值命名為有「’」的「A’」，而沒有使用到斜線標記的真值命名為沒有「’」的「A」。

「%{A}」中，{A} 指的是含有沒有使用到斜線標記的真值 A 的域，因此，%{A} 指的就是 {A} 的抽象，它同樣是一個域的集合。「b」，也就是「{A’}」，是這樣一個域的集合中的其中一個元素，因此，我們可以得到「({b} = {A’}) ∈ %{A}」。

如果要標示出域中的實體哪些使用到斜線標記，可以將 % 移到域的後方：

{A’} ∈ {A}%(<B>, <C>, …)

一個例子是：

{1/人.昨天.早上.去.2/地方} ∈ {<小明#1>.昨天.早上.去.<學校#2>}%(<小明#1>, <學校#2>)

上面的例子中，{小明.昨天.早上.去.學校} 對 (<小明#1>, <學校#2>) 取抽象後，其中一種結果是 {1/人.昨天.早上.去.2/地方}。(<小明#1>, <學校#2>) 使用到的是「T 系統」一章中介紹過的尖括弧與井字標記。

結合使用@符號的域前綴，我們也可以得到如下的式子：

(@{T_1, P_1}{A})%(T_1, P_1) = @{…%{T_1}, …%{P_1}}{A}

比如：

A=小美.今天.早上.在.學校.對面.的.早餐.店.吃.早餐

@{T_1, P_1}{A}=@{今天.早上, 在.學校.對面.的.早餐.店}{小美.吃.早餐}

(@{T_1, P_1}{A})%(T_1, P_1)

= (@{今天.早上, 在.學校.對面.的.早餐.店}{小美.吃.早餐})%{今天.早上, 在.學校.對面.的.早餐.店}

= @{…%{今天.早上}, …%{在.學校.對面.的.早餐.店}}{小美.吃.早餐}

= @{…%{T_1}, …%{P_1}}{小美.吃.早餐}

所以，「(@{T_1, P_1}{A}%(T_1, P_1)」指的就是：將「{小美.今天.早上.在.學校.對面.的.早餐.店.吃.早餐}」中的時間、地點先移到前綴當中之後，再對前綴裡的元素取抽象形成的抽象域。

注意到，因為定義上我們只對域取抽象，而不對語元取抽象，所以先把 T_1 和 P_1 放到域中，形成 {T_1}、{P_1} 後，先取抽象再展開（…%{T_1}、…%{P_1}），其中「…」是「域的表記方式」一節中介紹過的展開符號，「{…{A}}={A}」。

－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－

我們另外引入「::」符號（兩個連續的冒號「:」）來表示實例：

b ∈ %a

a ∈ ::b

比如：

a={小狗.在.公園.裡.玩.球}

b={1/動物.在.2/地方.玩.球} ∈ %{小狗.在.公園.裡.玩.球}

a={小狗.在.公園.裡.玩.球} ∈ ::b

上面的例子中，b 是 {1/動物.在.2/地方.玩.球}，它是 {小狗.在.公園.裡.玩.球} 的一個抽象，因此 b ∈ %a。

::b 是 b 這個抽象域的實例集合，由於「/動物」有很多種合法的代入詞組，「/地方」也有很多種合法的代入詞組，所以 ::b 是一個域的集合。a 是 ::b 這個「域的集合」當中的一種，因此 a ∈ ::b。

另外，我們也可以得到：

a ∈ ::(%a)

因為 a 取抽象後得到的抽象域，它的實例集合裡，一定有 a 本身。

－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－

我們使用「#」符號來表示域的「要件」。

#a 是一個「域的集合」，每個 #a 中的元素都是 a 的一個子集。

{c}{

大型.犬.的.體型.比.中型.犬.大,

大型.犬.的.食量.比.中型.犬.大,

大型.犬.需要.的.空間.比.中型.犬.需要.的.空間.大

}

上面的這個域 c 可以有一個 $d_1 \in \#c$：

{d_1}{

/大型_犬,

/中型_犬

}

但是觀察到，c 當中沒有 d_1 的元素「/大型_犬」與「/中型_犬」，這是由於我們使用了比較簡便的形式來表示 c。我們也可以將 c 寫成：

{c}{

/大型_犬.的.體型.比. /中型_犬.大,

/大型_犬.的.食量.比. /中型_犬.大,

/大型_犬.需要.的.空間.比. /中型_犬.需要.的.空間.大,

/大型_犬,

/中型_犬,

}

而且根據存在假設，c 域中用到了透過斜線標示的元素「/大型_犬」與「/中型_犬」，就表示在 c 域的範疇中，「/大型_犬」與「/中型_犬」存在（因為若非如此，c 中的某些真值無法產生意義）。

這樣，我們就可以看出 $d_1 \in \#c$，即要件域 d_1 是 c 的一個子集。

d_2、d_3 也各自是 c 的一個子集：

{d_2}{ /大型_犬 }

{d_3}{

/大型_犬,

體型

}

並非所有子集都是域的要件，對於一個域來說，我們可以把其中特定的元素聯繫起來形成一些「要件域」，這些域就是域的要件，而其它的子集不是，比如在 c 中：

{c}{

大型.犬.的.體型.比.中型.犬.大

大型.犬.的.食量.比.中型.犬.大

大型.犬.需要.的.空間.比.中型.犬.需要.的.空間.大

{d_1}{/大型_犬, /中型_犬}

{d_2}{/大型_犬}

{d_3}{/大型_犬, 體型}

}

所以，「要件域」裡的元素就是域的「要件」，「要件域」必定是域的「子域」，但不是所有域中的子域都是要件域，也不是所有域中元素都會在某個要件域當中，下一節裡，我們會看到取要件的用處為何。

－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－

另外，看到下面幾個域之間的關係：

{d_4}{小明.養.的.巨型.貴賓}

d_2 ={/大型_犬} ∈ #c

$d_4 \in ::d_2$

因為 $d_4 \in ::d_2$，即 d_4 是 d_2 的一個實例，而 d_2 同時也是 c 的要件域，所以我們稱 d_4 為 c 的一個「要件實例」。

域 a 透過抽象過程而符合另一個域 b 的要件，或者說是判斷「一個域 a 是另一個域 b 的要件實例」的過程，是一種「要件涵攝」，在這裡，也可以說是把域 a 涵攝於域 b。

6.3 注意力域與啟動條件

由於系統的推論資源有限，受到單步推論需時、推論帶寬、記憶容量等的限制，因此要形成完整的推論鏈，系統只能關注存在於注意力域當中的真值。

注意力域是系統表示域的一個子域，真值可以由表示域向內進入到注意力域內，也會由注意力域當中移出。系統來自外部輸入的資訊，會經過篩選之後進入到注意力域當中，不過注意力域當中的內容除了來自外界輸入以外，也可能是要件被啟動的域裡的其他真值，或者推論鏈建構過程當中所使用到的真值。

－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－

看到下面這樣的域：

```
{a}          //過.馬路
{
    過.馬路.時.要.看.行人.號誌.燈,
    綠燈.時.可以.過.馬路,
    紅燈.時.不能.過.馬路,
    過.馬路.時.要.注意.兩側.的.來車
}
```

這是系統將與「過馬路」相關的一些真值聚集在一個域當中。系統可能將「過.馬路」或者「<我>.在.過.馬路」作為這個域的其中一個要件域（如果像這裡一樣，發現「<我>.在.過.馬路」沒有在上面寫出的 a 域當中，我們卻說它在 a 的某個要件域當中，那麼由於要件域是域的子域，所以事實上，討論中的真值也是在域當中，只是為了篇幅考量省略未寫出）。當注意力域當中的真值涵蓋了要件域當中的所有真值，也可以說是注意力域中的元素「覆蓋」了某個域的「啟動要件域」時，這個域就被啟動，使得更多的域中真值進入到系統的注意力域當中。特別注意到，一個域可以同時有很多要件域，也可以同時有很多啟

動要件域，只要其中一個被注意力域覆蓋，域就會啟動。

再看到一個例子：

{b}　　//買.東西.選擇.大.包裝.或.小.包裝

{

大.包裝.的.商品.裡.量.比較.多,

小.包裝.的.商品.裡.量.比較.少,

大.包裝.的.商品.單價.可能.比較.低,

如果.大.包裝.的.商品.單價.比.小.包裝.的.商品.低.，.買.大.包裝.的.商品,

如果.大.包裝.的.商品.單價.比.小.包裝.的.商品.高.，.買.小.包裝.的.商品,

如果.大.包裝.的.商品.單價.和.小.包裝.的.商品.一樣.，.但是.商品.保存.期限.短.，.買.小.包裝.的.商品,

如果.大.包裝.的.商品.單價.和.小.包裝.的.商品.一樣.，.而且.商品.保存.期限.不短.，.買.大.包裝.的.商品

}

系統可能將「商品」、「大.包裝」、「小.包裝」作為這個域的其中一個啟動要件域的元素，成為這個域的啟動條件。當系統遇到類似的狀態，即選擇應該購買大包裝還是小包裝的商品時，這個域中的真值就會進入到系統的注意力域當中。

啟動條件是不完全系統在注意力域空間有限、推論能力也有限的情況下，形成的一種必要機制，它使得系統可以在接收到輸入時（某個給定時段內的輸入經過篩選後，資訊量是有限的），快速將透過經驗或學習而來，與當前輸入有關的真值也納入到注意力域當中。這樣一來，系統就不需要進行我們討論過的完全系統的推論方式，即把整個表示域當中所有的規則前提都掃描過，並進行所有可以發生的推論，而是只針對「注意到」的部分進行推論。

由於注意力域的空間有限，當新的輸入資訊或推論鏈使用到的真值不斷進入到注意力域當中，已經存在在當中的一些真值需要被排除。在生物當中，這可能會在細胞電流的強度降到低於閾值時自然發生，但在人工智能系統當中，我們需要特別設計一個方式來決定移除注意力域中的哪一些真值，以使當中的真值數量維持在一個範圍以內。

－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－

本節強調的是，注意力域中的元素覆蓋某個域的啟動要件域時，會啟動域中的推論規則，且自啟動要件域的範圍往外，會有更多域中的元素進入到注意力域內。簡而言之，一個域中的啟動要件域「成為注意力域的子域」，就是這個域裡的推論規則「進入系統的注意力域中」並「啟動」的條件。

在一些域當中，啟動要件域週邊（即推論距離近）的真值與推論結果，會滿足其它域的啟動條件，導致注意力域擴散到其它域當中。如果推論帶寬不足以支撐兩個域中的推論同時進行，注意力域有可能會轉而開始進行後來擴散到的這個域中的推論，而停止前一個域的推論。

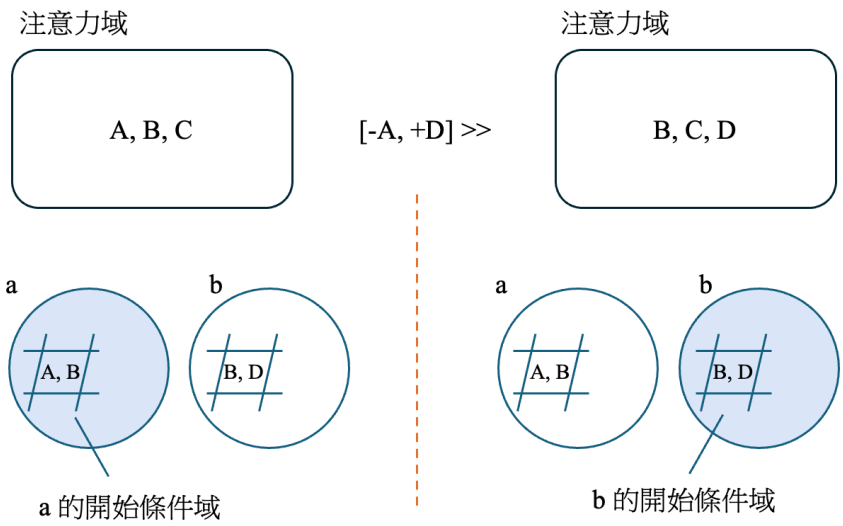
6.4 推論的發生

推論起始於「開始域」，並且經過一連串的步驟後，可能抵達完成域。

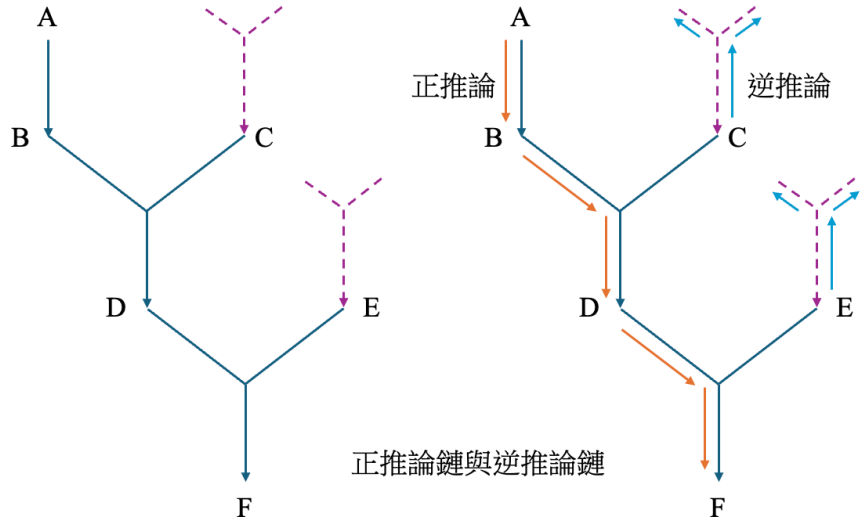
完成域是系統認為這個推論「完成」的狀態，其內有完成條件，比如當我們思考「某個家具是否適合擺在自己家裡」，而達到「適合」或「不適合」的結論時，即到達完成域。不是所有經由推論定形建構起的推論鏈都會抵達完成域，許多時候，由於所需資訊缺乏或推論時間有限等原因，推論鏈會停在未滿足完成條件的地方，但是已建構的部分仍會使系統當中的真值增加。

開始條件是開始域的一個內部域，且通常是「抽象域」，比如開始條件可能是：{1/物.的.速率.是.多少.?}，完成域中的完成條件可能是：{1/物.的.速率.是.每秒.2/數字.公里}。

當注意力域涵蓋的內容是開始條件的超域時，也就是說，注意力域當中出現了「1/物.的.速率.是.多少.?」時，開始域被啟動，並開始建構推論鏈以期在接下來的一段時間內，注意力域當中出現「1/物.的.速率.是.每秒.2/數字.公里」，滿足完成條件而抵達完成域。



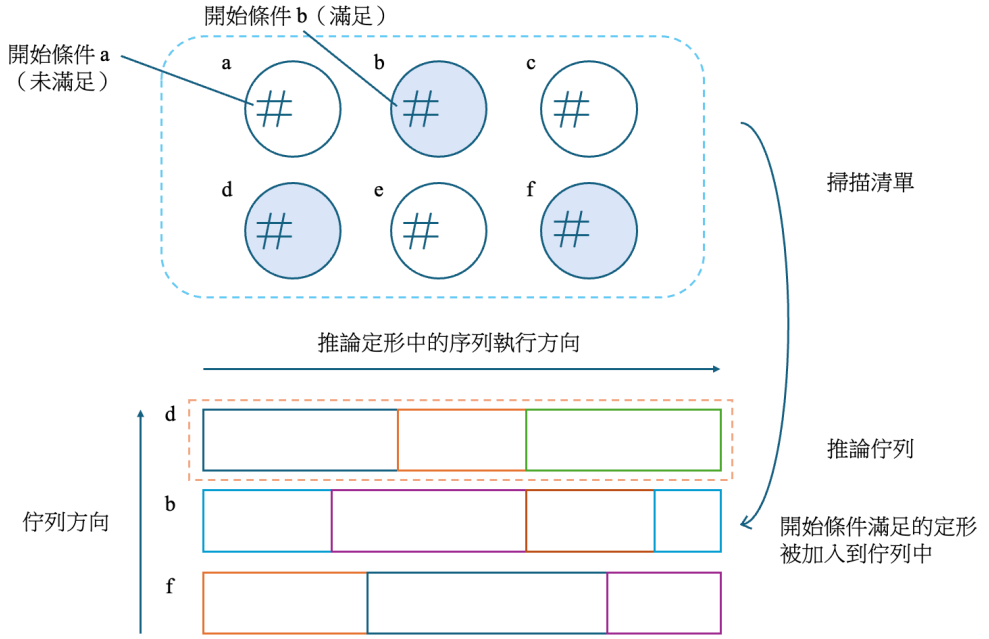
上圖中，注意力域為 {A, B, C} 時，a 的開始條件滿足，b 的開始條件不滿足，注意力域變為 {B, C, D} 時，a 的開始條件不滿足，b 的開始條件滿足。



由開始域到達完成域的推論鏈是「正推論鏈」，比如上圖當中，先由真值 A 得到 B，又由 B^C 得到 D，依此類推，直到得到 F。不過，有時候推論鏈上的真值不在系統表示域中，比如由 A 得到 B 後，雖然知道 B^C 可以得到 D，但是 C 不在域中（而 ～C 也不在域中）。

這時，系統可以尋找以 C 作為完成條件的「論形」，並且試圖從該論形的鏈上某個所有前提都已被滿足的節點往下推論，直到得到 C，使得原來進行的論形當中，因為 B 與 C 此時都被滿足，可以由 B^C 繼續往下推論至 D，而此時，又知道雖然 D^E 可以得到 F，但是 E 不在域中，因此另外又構造逆推論鏈，依此類推。

論形由「正推論鏈」與「逆推論鏈」共同組成，論形的正推論鏈上有一些入口，當其中一個入口域的開始條件被滿足，就可能開始畫出這個圖形。另外，給定開始條件時，正逆推論鏈時常會共同依據固定的順序，經過同樣的一組節點，並重複畫出一樣的形狀，我們也把這種重複畫出的形狀叫做「推論定形」。



系統當中的開始域共同構成一個掃描清單，當注意力域當中含有的真值改變，使得一些開始域的開始條件被滿足時（比如上圖中圓圈內塗淺色者），這個推論定形對應的推論序列（包含正推論鏈與可能所需的逆推論鏈）被加入到推論佇列當中。推論佇列當中，越上方的推論序列會越優先進行，直到完成條件滿足而被從佇列當中移除，或者原本在下方的佇列被排到上方而「插隊」進行。

「插隊」發生的原因可能是系統的注意力域當中真值又改變，使得系統重新「注意」到某個先前進行到一半的推論序列。新注意到的推論序列是否應該被加入或提高到序列最上端，還是應該依照某種規則而被排入到佇列的（垂直）中間，屬於系統最佳化的研究範疇。

系統並不是無時無刻都在處理推論佇列當中的推論，當系統的「行動掃描清單」當中有需要避免的負面狀態、即將發生的負面狀態、可追求的正面狀態等，而佔據系統的推論頻寬時，系統可能不會處理推論佇列當中的序列。從這個角度來看，可以把「進行推論」看作是行動的一種，與其它系統會執行的行動一起共享一個佇列，有關行動掃描清單與行動的安排，請見「行動」章節。

6.5 無限推論問題

高階智能系統理論發展的初期，因為尚未有「不完全系統」的概念，在分析系統處理「與迴圈、週期與順序相關的推論」的行為時，遇到了相當關鍵的「無限推論」問題。

{a_1} [規則 R_1]

If B^C

-> ～D

B 1/地方.有.2/數字.3/單位.4/物

C 2/數字.不.等於.5/數字

D 1/地方.有.5/數字.3/單位.4/物

上方的規則在「兩種推論方式」一節中曾討論過。在完全系統的表示域當中，如果存在以上的規則 R_A，以及無限多對互不相等的正整數對，給定前提 B 成立，會有無限多組正整數對使得 C 成立，並推論出 ～D，即 {a_1}{R_A, B, C} -> {a_2}{R_A, B, C, D}。

這樣一來，如果系統必須花費時間才能處理一個推論，且同時能夠處理的推論數量有限，整個系統在其存在時，理應會不間斷地處理應用這個規則的推論。然而，透過觀察自己作為一個智能系統的行為，R.K.C 認為雖然規則 R_1 存在，且當 B 在表示域中時，雖然確實系統中有無限多組正整數對使得 C 成立，系統卻沒有無時無刻地應用規則 R_1 進行推論，整個系統當中，也並未存在無限多個形如 ～D 的真值。

乍看之下，「無限推論」問題似乎使得要不是「規則的前提都成立時，就會導出規則的結論」這個法則不成立，要不就是前段中提到的觀察不成立。然而，「根據規則進行推論」作為高階方法的基礎，是無論如何不能放棄的，不過，系統沒有無止盡的帶寬和記憶空間，似乎也是顯而易見。

事實上，真正不成立的假設是「系統在規則的前提都成立時，就會「立刻」自動導出規則的結論」，關鍵不在於放棄根據規則的推論，而是不假設系統隨時都會進行所有能夠進行的推論。不過，既然認為系統不會無限制地構造正推論鏈，那麼系統是否與何時會構造當前可構造的正推論鏈，就成為需要著手解決的問題。

在這樣的背景下，必須研究不完全系統的特性，並在此基礎上繼續建構理論。本節當中，我們利用前面介紹過的注意力域與啟動條件概念，來解釋不完全系統「是否與何時」會進行這類與「迴圈、週期與順序」有關的推論。

比如，我們可以在 a_1 上加上啟動要件域：

{a_2=#a_1}{

B=1/地方.有.2/數字.3/單位.4/物,

C=2/數字.不.等於.5/數字,

D=1/地方.有.5/數字.3/單位.4/物

}

根據這個設計，只有當注意力域中同時存在 B、C、D，即同一個地方、某物數量不同時，才會啟動 a_1，並根據 R_1 推導出矛盾的結果，而若存在注意力域中 C 不成立，或者「只有 B、沒有 D」、「只有 D、沒有 B」等情形，就不會啟動 a_1 中的推論過程。

－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－

{b_1} [規則 R_2]

If E^F

-> G

E 1/一天.是.星期.2/星期數字

F 星期.2/星期數字.的.前.一天.是.星期.3/星期數字

G 1/一天.的.前.一天.是.星期.3/星期數字

上面的規則顯示，我們可以由「今年.的.端午.節.是.星期.五」和「星期.五.的.前.一天.是.星期.四」推論出「今年.的.端午.節.的.前.一天.是.星期.四」。一般來說，我們可以假設系統中有以下的真值：

星期.一.的.前.一天.是.星期.日 H_1

星期.二.的.前.一天.是.星期.一 H_2

星期.三.的.前.一天.是.星期.二 H_3

星期.四.的.前.一天.是.星期.三　　H_4

星期.五.的.前.一天.是.星期.四　　H_5

星期.六.的.前.一天.是.星期.五　　H_6

星期.日.的.前.一天.是.星期.六　　H_7

這樣一來，只要知道任何「1/一天.是.星期.2/星期數字」，並定義其中「/星期數字」是「日」或「一」、「二」、「三」、「四」、「五」、「六」，就可以根據 R_2 推論出前一天是星期幾。

但是，應用 R_2 進行正推論會遇到無限推論問題，因為「今年.的.端午.節.的.前.一天.是.星期.四」一樣可以寫成「1/一天.是.星期.2/星期數字」，這樣一來又會依序推論出「今年.的.端午.節.的.前.一天.的.前.一天.是.星期.三」、「今年.的.端午.節.的.前.一天.的.前.一天.的.前.一天.是.星期.二」、…。雖然推論的結果本身並無錯誤，但是系統既無必要進行這樣的推論，觀察一般的智能系統的行為，也不會進行這樣的推論。

要解釋智能系統的行為，我們可以將這些真值與規則封裝起來，即構造：

$\{b_1\}\{ R_B,$

$H_1, H_2, H_3, H_4, H_5, H_6, H_7 \}$

並且設定表示域 b_1 的啟動要件域：

$\{b_2=\#b_1\}\{$

E'=1/一天.是.星期.2/星期數字,

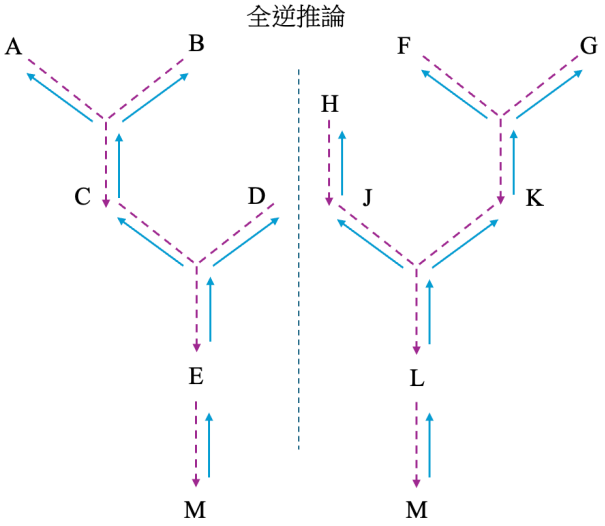
?(G'=1/一天.的前.一天.是.星期.2/星期數字),

}

其中，「?(G')」的「?」是推論鏈上的「鉤子」，隱含了「接下來的推論步驟當中需要系統中存在某個 G' 的實例」，及「現在系統中不存在那個 G' 的實例」的意思。只有當注意力域中含有 b_1 的啟動要件時，推論才會進行。

－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－

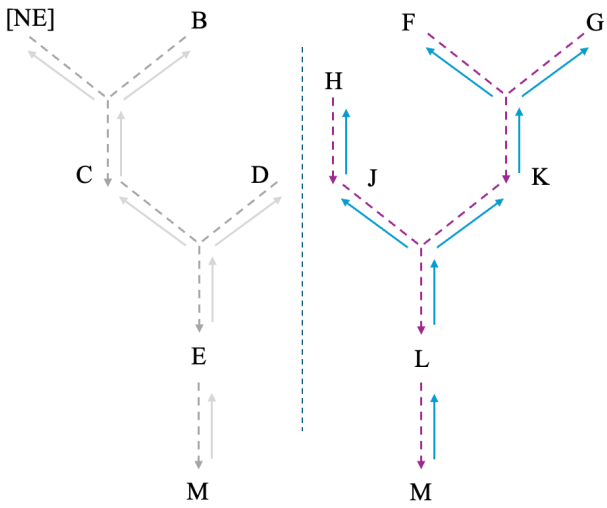
之所以要有「鉤子」的設計，是讓啟動要件域的概念除了可以用在「系統注意到一些事」所以開始進行某些推論以外，還可以用在「根據經驗，系統發現注意到某些真值，有助於正在進行的推論」這種情形上，這有助於系統以快速、省能量的方式來完成一些推論。現在，我們來深入了解這種情形。



在「推論的發生」一節當中，我們介紹過正推論鏈與逆推論鏈共同組成了論形。在需要構造逆推論鏈的節點上，如果當前系統中並非已經存在該真值，就會出現系統「先知道要推論出什麼」再尋找可以導出這個真值的方式的情形。

完全依靠構造逆推論鏈來試圖確定真值是否可導出的方式，叫做「全逆推論」，比如上圖當中，系統試圖使用全逆推論來構造出支持 M 的推論鏈，虛線隔開的兩次各自都是可以透過逆推論得到 M 的論形。如果 A、B、D 都在系統的表示域中，或者 F、G、H 都在系統的表示域中，那麼 M 就會得到支持。

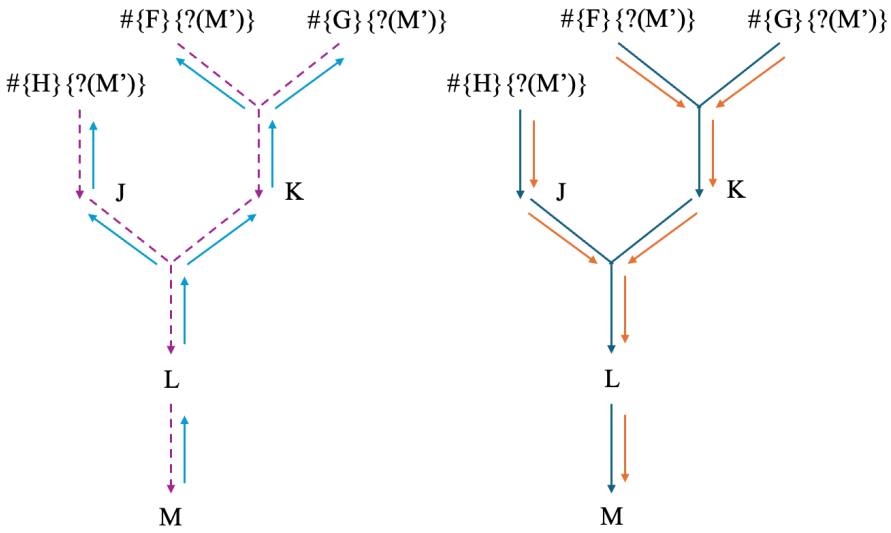
但是，有時候這種全逆推論是相當沒有效率的：



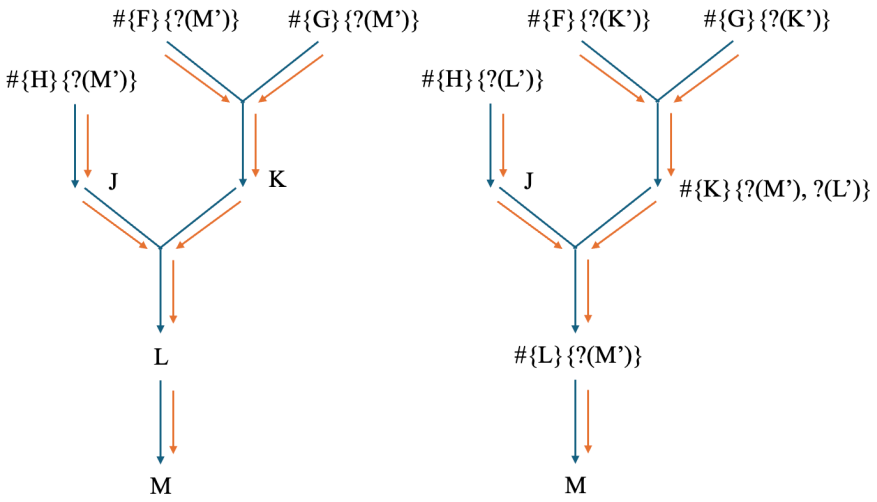
如上圖所示，系統可能從 M 出發進行逆推論，並從 E 節點一路往上試圖構造支持 M 的推論鏈，然而，到了 A^B->C 這個地方，才發現 [NE]A，也就是系統的表示域中沒有 A，而且也無法透過再往上構造逆推論鏈來推論出 A（得到這個結論，可能是因為給定的門檻時間內無法構造出以 A 為目標的逆推論鏈），這時，整個包含 E->M、C^D->E、A^B->C 的這個構造逆推論鏈的嘗試就徒勞無功了，雖然這次嘗試當中或許使表示域中增加了一些真值，但是因為系統的記憶空間有限，這些真值或許還要被移除，使得整個過程不僅沒有什麼幫助，還有可能延誤到系統進行正確的、含有 F、G、H 等節點的推論。

因此，當系統成功透過構造出含有 F、G、H 等節點的逆推論鏈以支持 M 後，可以透過設定一些鉤子，加速日後構造出支持形如 M 的真值的推論鏈的速度。

比如看到下圖：



圖中左側是原本的逆推論過程，經過成功的鏈構造過程後，我們可以在含有 F、G、H 的域上加上啟動條件，並放入 ?(M') 元素。如同前面提過的，?(M') 的意思是「接下來的步驟當中需要系統中存在某個 M' 的實例」，及「現在系統中不存在那個 M' 的實例」的意思，所以，如果正在進行需要得到 M 的推論，而使 M 進入到注意力域的話，透過這些鉤子，可以同時使 F、G、H 進入到注意力域，這時，就可以很快的構造出正推論鏈來支持 M，而不需要進行耗時的逆推論鏈構造過程了。

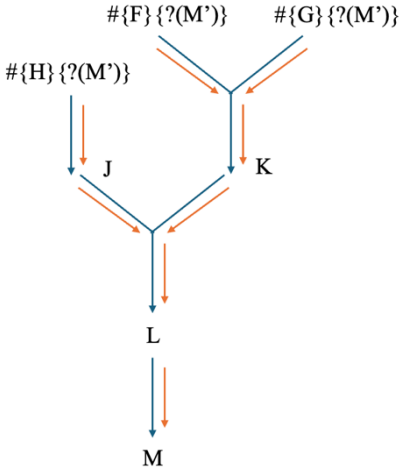


另外，這種鉤子的佈置方式不是唯一的，如同上圖所示，我們可以在有 F、G、H 的域上加上 ?(M) 鉤子，也可以轉而在不同的節點上放上一些包含鏈的後方節點的鉤子。像右圖的這種佈置方式，系統注意到 M 時，會再注意到 L 與 K，注意到 L 時，會再注意到 H 與 K，注意到 K 時，會再注意到 F 與 G，而如果有一些鏈上需要的節點沒有被鉤住，則從該節點向上一樣會進行逆推論過程。像這樣，系統會分階段的透過構造小段的正推論鏈與逆推論鏈，陸續把整個所需的論形構造出來。

6.6 範式與路徑依賴

範式是一系列互有關連的推論規則，其中可能包括複數個正推論鏈與逆推論鏈上的節點，與論形不同的是，它只包含了這些節點，而並未指出它們之間的連結關係。

我們先針對幾個常見的概念進行比較與討論以避免混淆：



以上一節帶鉤子的推論鏈為例：

- 論形：指的是正推論鏈與逆推論鏈共同構成的形狀，它畫出了表示域中的哪些真值經由什麼的過程得到結論
- 定形：根據給定的開始條件，重複畫出的論形
- 定式：指的是系統用以快速完成推論的固定順序，比如，在上圖中，「推論出 M 的定式」可能是「第一步：F^G->K」、「第二步：H->J」、「第三步：J^K->L」、「第四步：L->M」。為了方便記憶與理解，我們也可以使用一個比喻，把它看作是電玩遊戲中的按照特定順序進行可以快速發揮出特定效果的「組合技」
- 範式：指的是一些可以共同推論出某些結論的真值集合，比如，在上圖中，「F、G、H」可以是一個範式，「F、G、J」也可以是一個範式，「F 到 M 全部」也可以是一個範式。它有點像是「組合技」當中常一起用到的一些「技能」或「招式」組合

由正推論開始時：

（1） 系統注意到某些真值，這些真值使得某個域被啟動，進行了某些推論

（2） 系統注意到剛才已執行的推論的結果，以及一些其它真值，這些真值加起來又使得另一個域被啟動，進行了另一些推論

（3） 經過一些步驟後，系統得到某個真值，這個真值在目前推論鏈的最下方

（4） 從正推論開始到目前為止，畫出的所有正推論鏈與逆推論鏈，共同組成了一個推論圖形，即「論形」

（5） 系統認為這個推論過程是重要的，比如推論出有用的結論，這時，設置鉤子以使未來要得到類似的結論時，能夠快速啟動中間經過的節點

（6） 在某個之後的時間點，系統需要得到類似形式的結論。這時，帶有鉤子的某些含有特定真值的域依照某個順序（比如與在論形中的位置有關，或者與推論距離有關）啟動，這個順序就是「定式」

（7） 按照這個定式的順序，畫出了與之前類似的論形，且避免了某些不必要的推論過程，這個會重複畫出的精簡圖形，就是「定形」

（8） 一些互有關聯的真值，時常共同出現在論形中，或者在定式中前後出現，就共同組成「範式」

由逆推論開始時：

（1） 系統需要得到某個結論時，先尋找以這個結論為推論結果的規則，並開始進行逆推論，要得到逆推論上的節點而需要用到更多真值時，就繼續進行逆推論

（2） 經過一些步驟後，系統找到某些節點，它們的前提全部存在表示域中，從這些節點的前提出發，可以向下畫出一個推論圖形，或「論形」，使得需要得到的結論位在圖形的最下方

（3） 系統認為這個推論過程是重要的，比如推論出的結論有用，這時，設置鉤子以使未來要得到類似的結論時，能夠快速啟動中間經過的節點

（4） 某個之後的時間點，系統需要得到類似形式的結論。這時，帶有鉤子的某些含有特定真值的域依照某個順序（比如與在論形中的位置有關，或者與推論距離有關）啟動，這個順序就是「定式」

（5） 按照這個定式的順序，畫出了與之前類似的論形，且避免了某些不必要的推論過程，這個會重複畫出的精簡圖形，就是「定形」

（6） 一些互有關聯的真值，時常共同出現在論形中，或者在定式中前後出現，就共同組成「範式」

－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－

範式之間，根據其所屬的系統任務類型，也可能互有類似之處。

比如，看到以下幾個可能的範式，為了不佔用太大的篇幅，當中的節點刻意包含的並不完整：

- 是否是健康的餐點（分類）

  {

  1/炸_的_食物.不.健康,

  1/均衡_的_餐點.比較.健康,

  1/食物.含有.我.的.身體.需要.的.營養素 -> 1/食物.是.健康.的.食物

  }

- 決定住哪家飯店（選擇）

  {

1/飯店.比.2/飯店.便宜 -> 應該.選擇.住.1/飯店,

2/飯店.的.設施.比.1/飯店.好 -> 應該.選擇.住.2/飯店,

2/飯店.比.1/飯店.的.網路.評價.高 -> 應該.選擇.住.2/飯店

}

- 安排工作的順序（安排行程）

{

1/工作.比.2/工作.緊急 -> 先.做.1/工作,

1/工作.比.2/工作.輕鬆 -> 先.做.1/工作,

2/工作.比.1/工作.重要 -> 先.做.2/工作,

1/工作.比.2/工作.緊急.但.比較.不.重要 -> 先.做.2/工作

}

像這樣，諸如「分類」、「選擇」、「安排行程」等常見的任務類型，可能各自會產生許多類似的範式，比如「決定住哪家飯店」和「決定吃哪家餐廳」，很可能有許多互有對應的規則。這時，也可以把範式再向上抽象後，經過添加、減少一些規則，與向下實例化的過程來「複用」到其它的任務上。

－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－

給定開始條件與完成條件，即使其中的路徑不唯一，一旦系統建構出開始域到完成域的推論鏈後，後續為了節省推論能量，就會有「路徑依賴」的傾向。

有時，初次建立起的推論鏈並未納入所有相關的真值，因而忽略了其它同樣也可能被推論出的結果，或者並非距離最短或最確定（有關確定度的討論，請見「學習」章節），然而，系統在後續進行同樣的任務時，仍然採用了同樣的論形。考慮到重新搜索與建構路徑將耗費的時間與運算資源，這可能是最佳化或非最佳化的結果：當「答案足夠好」、「推論過程足夠穩健」時，得到「不是經過最全面考量的答案」或者進行「不是最短路徑的推論」可能是適當的作法。

不過，由於系統透過學習過程，會不斷添加新的真值與更新既有真值的確定度，有時候依賴的路徑可能變得不可靠，甚至斷裂（比如必要節點不再是表示域中的真值，而不應再畫出同一個論形），此時，必須考慮重新找尋新的路徑。

這種路徑依賴的情形存在於系統所進行的大部分任務當中，無論是否涉及身體的控制皆然，比如「即使已經開放了新的道路，仍然繼續走之前習慣的路線」、「雖然有更專門的網站，仍然去自己習慣查找資料的網站搜尋」、「明明已經知道這個飯店的水龍頭冷熱和自己習慣的旋轉方向不一樣，仍然不時轉錯」等等情形。

這些情形有時候是權衡的結果，比如根據經驗，可能這樣做的成功率比較高，有時則只是因為系統使用到不適合的真值或規則，這可能是因為「鉤子」鉤住了這些真值，導致它們在推論過程中，比更適用的真值更早出現，這種情形，以人類來說，在急躁（推論時間短）或者精神不佳（推論能量不足）的時候尤其容易發生。

路徑依賴減輕系統推論負擔的表現也可以透過「阻止系統跟隨依賴的路徑」來觀察：比如，一個擅長德州撲克的選手，如果今天玩的是規則幾乎完全相同，只有牌的大小完全倒過來（原本是 A 最大、K 次之，Q 再次之，依此類推，2 最小，變成 2 最大、3 次之、4 再次之，依此類推，A 最小）的比賽，可能做出同樣品質的結論所需的耗時會明顯延長，但是對於完全沒有玩過任何撲克遊戲的人來說，牌的大小是否反過來，對他的思考需時可能就沒什麼影響。

另一個例子是，如果今天要求一個人用數十色的蠟筆在著色本上著色，即使未要求需要按照經驗著色，他可能仍然會將樹葉塗成綠色、葡萄塗成紫色、大象塗成灰色，然而，如果要求他「隨機」塗色填滿這些圖形，反而可能會耗費更長的思考時間，但如果是一個尚且不諳世事的幼童來執行這個隨機塗色工作，並假設塗色動作本身的熟練度與速度並無不同，則可能反而完成的更快。

由於路徑依賴雖然可能降低推論結果品質，卻可以有效縮短所需的推論耗時，在設計人工智能系統時，也必須考慮納入類似的機制。

6.7 推論距離

推論距離的概念可由系統的推論速率（前面提過，為系統「同時執行的推論鏈數量」x「單位時長內每推論鏈推論步數」）與推論耗時推導出來：

速率 = 距離 ÷ 時間

距離 = 速率 x 時間

推論距離 = 推論速率 x 推論耗時

在不考慮系統推論速率隨著時間變化，以及透過多線程並行處理論形不同部分的前提下，由開始條件至完成條件的推論距離就是「系統的推論速率 x 推論耗費的時間」。

我們通常只比較兩個推論過程在同一個系統當中的推論距離，而不比較不同系統進行同一個推論過程的距離差異。此時，將這個系統的推論速率視為固定，即不需考慮推論速率的定義中含有推論步數的部分，且前面也提到不考慮多線程並行，這時，我們就可發現推論距離只與推論耗時成正比。推論距離沒有單位、只比較相對大小，之所以不將推論距離的單位設為「幾步」，是因為推論過程還涉及比對啟動要件域、消解矛盾等等步數沒有嚴格定義的程序。

簡單來說，對於同一個系統而言，能夠「更快想出來」的推論，它的推論距離比較短，要「想比較久才能想出來」的推論，它的推論距離比較長，比如下面的例子：

下列三個一組的詞組中，哪兩個是一對，另一個與其它兩個不是一對？

- 紅茶、鮮奶茶、拿鐵
- 小客車、公車、火車

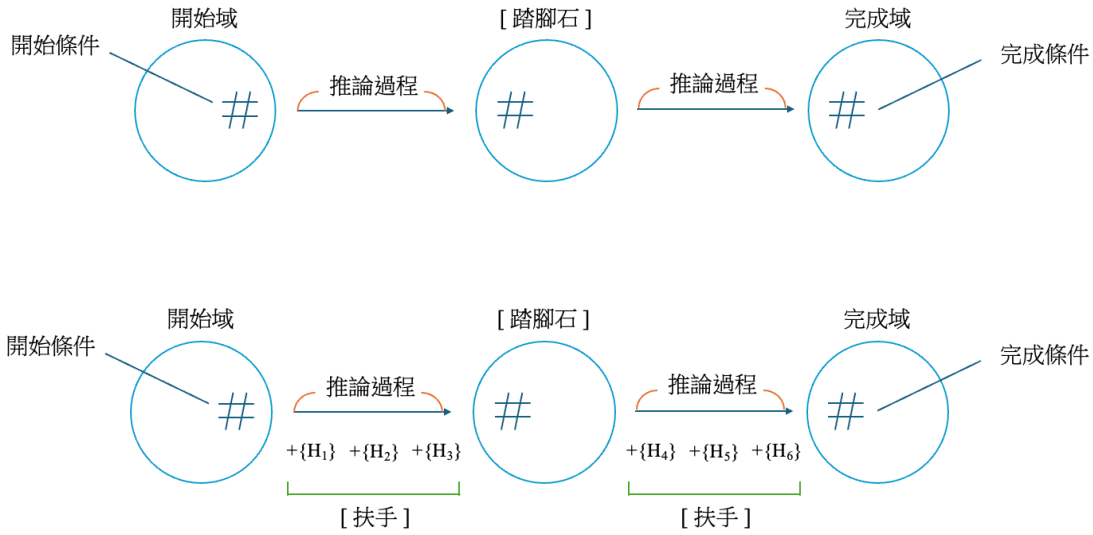
這是一個常見的「定義不精準」、「答案眾說紛紜」的遊戲，但我們可以從中窺見推論距離的用處。所謂「兩個詞組是一對」，意思是兩個詞組之間有某種關係，而「另一個與其它兩個不是一對」，意思是第三個詞組與前兩個詞組之間，這種關係都不成立。

在第一個例子當中，如果認為「紅茶和鮮奶茶」是一對，拿鐵與它們不是一對，可能是因為「紅茶和鮮奶茶都是茶」，而拿鐵不是，如果認為「鮮奶茶和拿鐵」是一對，可能是因為「鮮奶茶和拿鐵都含有牛奶」，而紅茶沒有。如果先想到一個成立的關係，就馬上回答，那麼回答的答案所對應的那個關係，有可能經過的推論距離更短，所以需時也短，但如果我們在回答前一下子就想到了複數個都成立而使答案不同的關係，因而覺得難以回答，這也可能代表推論出這些關係各自需要經過的距離差距不大。

類似的，第二個例子當中，也可能得出不同答案，即使都認為「公車和火車」是一組，也有可能是因為「都是大眾交通工具」，或者「都有車站」，具體給出答案的原因是兩者中何者，還是是給出其它原因，都會受到推論距離的影響。

值得注意的是，不同的系統記憶機制不同。在某些系統當中，要完成一個推論過程，需要將所經過的推論鏈的足夠多部分保留在工作記憶（系統記憶機制當中內容頻繁變更的一個部分）當中，如果工作記憶當中無法同時容納所需的部分，推論就有可能會「斷鏈」。一個直觀的例子是未經過特殊訓練的人類個體，如果在腦中試圖進行五位數乘以五位數的直式乘法，就可能會因為斷鏈而無法得出結果。

要有效克服斷鏈的情形，而構造出非常長的推論鏈，我們可以在推論過程當中設置「踏腳石」與「扶手」來協助完成整個構造。



上面的圖示當中，我們可以看到設置「踏腳石」就是在從開始域到完成域的整段推論過程當中，選取一些關鍵的節點，將其設定為推論的中間目標，以分段完成整段推論，或者把它「記下來」，以代表推論過程曾經到達過這個節點。

「記下來」的方式，比如我們在長時間思考時可能會把一些重要結論寫在筆記本上，下次看到時，就知道已經到達（想出）過這個節點上的真值。我們也可以把到達踏腳石為止的論形上所有需要用到的前提作為前提，把踏腳石作為結論，形成一個新的合成規則，這樣一來，下次推論時使用這個規則，所需時間就會大幅減少。

更進一步，我們還可以在推論過程上設置許多「扶手」，這些扶手不是「重要的中間結論」，而是推論過程當中使用到的一些真值，也就是形成那個推論圖形的「提示」。它的作用方式像是前面提到過的鉤子，可以使某些真值進入到注意力域當中，減少推論圖形上必須經過的無效分支，並快速分段完成整個推論過

程。要設置扶手，一樣可以透過紀錄的方式，將推論過程當中曾經想到的有關真值與實體、概念中介寫在筆記本等媒介上，以在後續推論時，輔助推論圖形的產生。

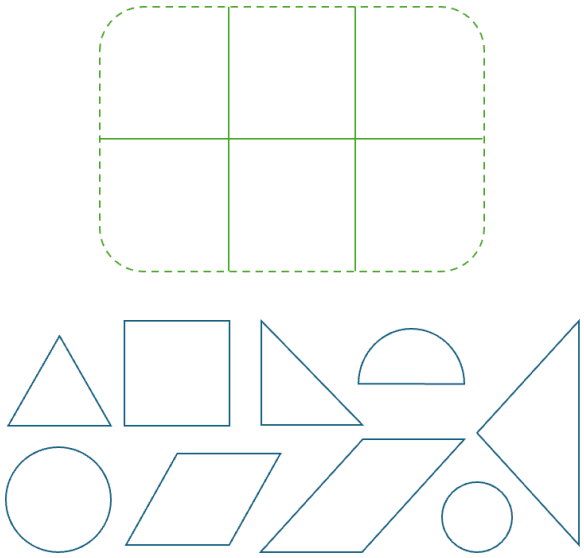
綜合來說，「踏腳石」使得系統一次只需處理較短的鏈，後續再把這些鏈連接起來即可（連接時，只需要記得每段鏈的前提與結論，不用記得中間過程）；「扶手」減少無效分支，使得鏈上非必要的分岔減少，這些方法都可以減少工作記憶有限，導致構造長推論鏈時容易斷鏈的問題。

另外，前面提過：

$$\text{推論距離} = \text{推論速率} \times \text{推論耗時}$$

因此，諸如使用合成規則、設置扶手來紀錄推論路徑等方法，因為可以在固定的系統推論速率下減少推論所需時間，我們也可以將它們視為縮短「推論距離」的方式。

6.8 有限記憶



在有限的記憶空間當中，系統必須決定留下哪一些真值以在節省空間的同時，仍然保留畫出特定推論圖形的能力；另一方面，由於記憶的儲存速率也有其上限，在大量資訊進入到系統當中時，系統也需要有一個機制來決定需要留下哪一些資訊，而篩除其它的資訊。

上圖當中，我們可以看到系統的記憶空間像是一個「工具箱」，這個工具箱有每格固定的大小，以及總共的上限格數，這使得為數眾多的如下方的幾何圖形（每個比喻了不同的真值）不能全部被放到工具箱內。如果這個工具箱的功用是讓我們可以取用內部的幾何圖形，以用筆沿著周圍在紙上畫出圖形，乍看之下，似乎因為沒有辦法容納所有工具，所以必須放棄畫出某些圖形的能力，不過，我們會發現，某些幾何圖形結合起來，可以與另一個幾何圖形的形狀相當；同時，有些幾何圖形大小太大，不能放到工具箱的格子內，必須被看作是數個其它圖形的組合，透過留下那些較小的成分圖形，來維持畫出這個較大圖形的能力。

如同這個「工具箱模型」所比喻的，系統的記憶空間有限，因此對於真值必須有所取捨，在空間仍然充裕時，我們可以簡單地留下所有的真值，但是已使用的空間接近上限時，就必須採用不同的手段進行「資訊壓縮」，也就是試著用不同的圖形來拼湊出某些其它圖形，避免同時儲存所有資訊而超過上限，導致新的真值無法再被存放，或者舊的真值以不理想的順序被排除。

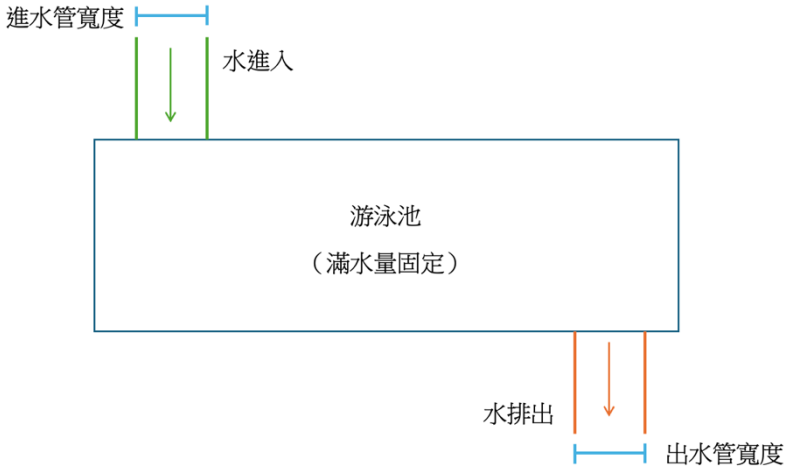
另外，在新真值透過資訊輸入的過程產生時，也必須考慮真值寫入的速率限制，這就像是有一條流水線上源源不斷的有新的幾何圖形經過，我們必須從中判別出哪些與工具箱內已有的圖形相同、哪些可以用已有的圖形拼湊出來、哪些似乎可以用以拼湊出更多的其它圖形等等，以決定有限的拿取速率下，應該取用哪些新的工具。

－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－

真值的佔用空間互相之間並不等同，並不是固定一個真值多少位元組，或者一個真值多少立方微米。事實上，由於上面提到的「資訊壓縮」的過程，一個真值對於特定系統在特定時間點的表示域而言的佔用空間大小，與對於其它系統、其它時間點的表示域而言，都可能有所異同。

要在這樣因素眾多的情況下估計真值的大小，可以使用「游泳池模型」:

系統的記憶空間像一個游泳池一樣，會有水的進入與水的排出，且滿水量是固定的。另外，由於水進入和排出的管道寬度也固定，所以一個時段內能夠進入到游泳池內的水量與離開游泳池的水量也有上限。



要得知真值對於一個系統當下的空間大小，可以把它當作游泳池的進水來看，如果我們讓進水管以最大速率進水：

飽和進水量 = 單位時間最大進水量 x 花費時間

而我們又假設系統的單位時間最大進水量不變，這樣，在這種飽和進水的狀態下，花費越長時間，進入的水量就越大，佔用的容量也越大，反之，花費越短時間，進入的水量就越少，佔用的容量越小。

這種評估方式無法讓我們得到真值佔用容量的絕對大小，但是可以得到相對大小，簡單來說，如果讓一個系統，比如人類個體，試著專注背起兩組資訊，那麼我們可以透過系統背起這兩組資訊各自的所需時間來衡量兩組資訊的相對佔用容量大小。

比如，先看到第一個例子：

大象、駱駝、綿羊、牧羊犬、公雞、天竺鼠、瓢蟲、螞蟻、綠藻

牧羊犬、瓢蟲、公雞、螞蟻、綠藻、綿羊、駱駝、天竺鼠、大象

這裡，要背起這些資訊，並按照順序複頌，除了需要記起哪些詞組有出現以外，還要記起它們之間的相對順序。對於第一組詞而言，這些出現的詞組指涉的實體對於充分認識這些實體體型大小的系統而言，是由大到小排序的，這時，系統只需要把詞組記起來，而不需要特別去記順序。因為在複頌時，即使忘記順序，只要記得出現了哪些詞組，並進行詞組指涉的實體大小的比較，就可以知道排序當中下一個是什麼。

但是，第二組詞的順序似乎很隨機，它不是以實體的尺寸大小排序，也不是昆蟲在前面、非昆蟲在後面，或者第一個字筆畫少的在前面，第一個字筆畫多的

在後面，因此，對於系統而言，很可能會需要花費更多的時間才能將它背起來（當然，我們假設的是系統沒有先背過第一個詞組，而是直接背第二個詞組）。如果我們真的觀察到這個現象，而且認為系統在背兩組詞組時的認真程度（是否飽和進水）相當，那麼，我們就可以判斷第一組詞組的佔用空間比較小，而且，這很可能是因為系統使用已經存在的內部真值來壓縮輸入資訊導致的。

這個資訊壓縮的過程發生在圖示裡面游泳池進水之前，因為游泳池指的是系統「記起來」的資訊，系統不是先記起來才決定要記哪些東西，而是先決定需要記哪些東西後，才經過這個進水的過程。

再看到另一個例子：

天空是藍的、草是綠的、蘋果是紅的、葡萄是紫的、蜘蛛有八隻腳、彩虹有七種顏色、星座有十二個、一小時有六十分鐘

天空是綠的、草是紫的、蘋果是藍的、葡萄是紅的、蜘蛛有六十隻腳、彩虹有八種顏色、星座有七個、一小時有十二分鐘

在這個例子當中，我們看到兩組的差別是對於大多數的人類個體來說，第一組當中的語句都是真的，只要記得比如「總共八個詞組」、「前面四個在講顏色」、「後面四個在講數字」、「前面四個是天空、草、蘋果、葡萄」、「後面四個是蜘蛛、彩虹、星座、一小時」這些資訊，就可以複頌出整組詞組。

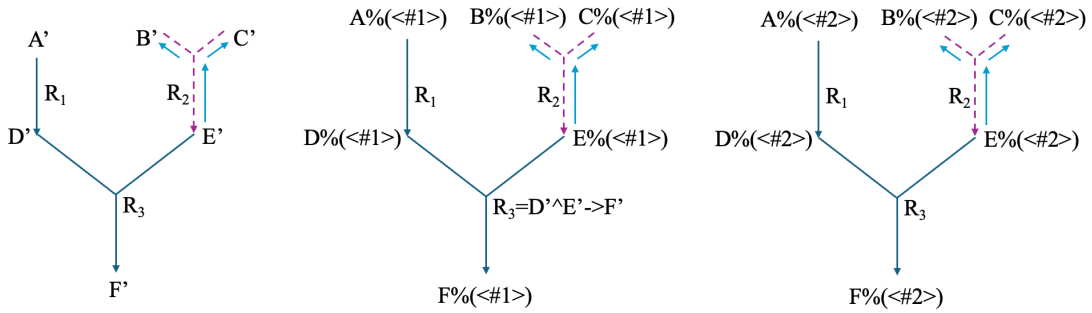
但是，後面一組詞雖然一樣「前面四個是天空、草、蘋果、葡萄」、「後面四個是蜘蛛、彩虹、星座、一小時」，但是每個詞組形成的語句對多數人類個體來說，都不在系統當中，甚至與既有的真值衝突。這樣一來，又要記起更多資訊，還要處理真值衝突的問題（當然，這種時候根據理論，可以另開一個域，給予不同前綴以避免混淆，但是對於人類這樣的個體來說，鉤子還是有可能先去鉤到不在這個域中的真值，或者要額外花費精力與時間避免發生穿透錯誤而導致不當的矛盾消解），因此，很可能實際上會需要花費更多的時間才能記起來。

如果真的觀察到這個現象，那麼，我們就可以判斷第一組詞組的佔用空間比較小，而且，這代表除了資訊壓縮的因素以外，我們還必須考慮真值衝突對於佔用容量的隱含影響。

－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－

回到選擇儲存哪些真值上的討論，系統必須決定要儲存哪些真值，才能夠使資

訊發生壓縮的過程是「無損」的，在無法達到無損的情況下，也必須追求損失程度較小。這個任務不僅對於我們前面看到的進水過程相當重要，在出水的過程當中也很關鍵，因為如果現在系統由於空間限制，必須選擇將某些真值排除（這與學習過程中因為矛盾消解導致某些真值被排除的情況不同），那麼選擇排除的真值，除非另有別的考量，否則應該是那些造成的資訊損失較小的真值。



看到上面的例子，最左側是一個推論圖形，右邊是這個圖形代入不同實例得到的結果。如果不能留下 A'、B'、C'、D'、E'、F'、R_1、R_2、R_3 全部的真值，我們必需留下哪些才能復現出這個推論圖形呢？

一種選擇是留下 R_1、R_2、R_3、A'、B'、C'。在 ?(F') 時，可以透過 R_3 構造逆推論鏈至前提 D'、E' 處，而 D' 再經過 R_1 到 A'，E' 再經過 R_2 到 B'、C'，這樣同樣的圖形就被畫出來了（雖然正推論轉為逆推論，但是完成這個逆推論後，可以設置鉤子重新轉為正推論），當然，另外再追加留下 D' 或者追加留下 E' 等情況，也可以畫出這個一樣的圖形。

不過，如果我們的目標不是複現出完全相同的推論圖形，而只是要能夠推論出 F' 就好，那麼，似乎可以只保留 R_3、D'、E'，這時，?(F') 經過 R_3 構造逆推論鏈至前提 D'、E' 處，推論就完成了，或者，只保留 R_1、R_3、A'、E'，?(F') 經過 R_3 構造逆推論鏈至前提 D'、E' 處，E' 已經存在，D' 經過 R_3 到 A'，這樣同樣可以使 F' 被推論出來。

綜合前面兩段的討論，我們會發現，雖然 R_1、R_2、R_3、A'、B'、C' 可以複現出整個圖形，但是再多保留 D'，或者再多保留 E' 的情況下，F' 都可以更快被推論出來，這告訴我們，在節省真值儲存空間的同時，可能會因為保留的真值太少，使得後續即使能推論出一樣的結果，但是耗時變長。當然，這也代表如果我們所設計出來的智能系統在儲存空間上幾乎不受到限制，它就可以透過儲存大量真值來避免需要重複構造推論鏈，並藉此加速推論，不過，這也可能

會使得它時常遇到真值衝突的問題，而必須不斷進行矛盾消解，有關這部分的機制，請見「學習」章節。

－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－

$\{A_1\} >> \{B_1\}$	$\{A_1'=A_1\%(<\#1>^1_2, <\#2>^1_5)\} >> \{B_1'=B_1\%(<\#1>^1_2, <\#2>^1_5)\}$
$\{A_2\} >> \{B_2\}$	$\{A_2'=A_2\%(<\#3>^1_2, <\#4>^1_5)\} >> \{B_2'=B_2\%(<\#3>^1_2, <\#4>^1_5)\}$
$\{A_3\} >> \{B_3\}$	$\{A_3'=A_3\%(<\#5>^1_2, <\#6>^1_5)\} >> \{B_3'=B_3\%(<\#5>^1_2, <\#6>^1_5)\}$
$\{A_4\} >> \{B_4\}$	$\{A_4'=A_4\%(<\#7>^1_2, <\#8>^1_5)\} >> \{B_4'=B_4\%(<\#7>^1_2, <\#8>^1_5)\}$
$\{A'\} >> \{B'\}$	$\{A'=A\%(<\#/N_1>^1_2, <\#/N_2>^1_5)\} >> \{B'=B\%(<\#/N_1>^1_2, <\#/N_2>^1_5)\}$

另一個層面的真值選擇問題發生在抽象的過程當中，如上所示，系統因為觀察到 $\{A_1\} >> \{B_1\}$、$\{A_2\} >> \{B_2\}$ 等過程，而將其抽象為 $\{A'\} >> \{B'\}$。右側，我們為了明確表示這個抽象過程是如何發生的，使用了比較完整的表記。

以 $\{A_1\} >> \{B_1\}$ 為例，這裡 $\{A_1'=A_1\%(<\#1>^1_2, <\#2>^1_5)\}$ 指的是 A_1' 是 $\{A_1\}$ 對 <#1>、<#2> 取抽象，其中 <#1> 的上標 1 是指域中的第一個真值語句，下標 2 是指這個真值語句中點號分隔出的第二個詞，<#2> 的上標 1 是指域中的第一個真值語句，下標 5 是指這個真值語句中點號分隔出的第五個詞，當然，在這裡，$\{A_1\}$ 當中只有一個真值語句而已。舉例來說，A_1 可能是「昨天.</小明#1>.搭車.到.<餐廳#2>.吃飯」，A_1' 可能是「昨天.</人>.搭車.到.</地點>.吃飯」。我們也可以使用「$A_1'=A_1\%(<\#1>^1_2, <\#2>^1_5)$」來作為「$\{A_1'\}=\{A_1\}\%(<\#1>^1_2, <\#2>^1_5)$」的簡寫，因為這裡域和真值不致於混淆。

我們要討論的真值選擇問題是，當儲存空間不夠時，是否應該要排除 $\{A_1\} >> \{B_1\}$、$\{A_2\} >> \{B_2\}$ 等，而只留下 $\{A'\}>>\{B'\}$ 呢？如果這樣做，理應能夠節省許多儲存空間，但是缺點是只保留了「重複因子」，而丟棄了「脫落因子」（見「抽象、實例與要件」一節），這時候，如果我們丟棄排除的是 $\{B_1\}$，但保留了 $\{A_1\}$，那麼根據同樣保留下來的 $\{A'\}>>\{B'\}$ 規則，可以重新再推論出 $\{B_1\}$，可是，這時候的 $\{B_1\}$ 是被推論出來的，而不是透過經驗得到，這種現象就是我們所常見的「記憶模糊」了，「好像應該是如何如何」的這種表述。因為脫落因子不見了，所以 $\{A_1, \ldots\{a\}\}$ 根據 $\{A_2\} >> \{B_2\}$ 得到 $\{B_1, \ldots\{a\}\}$ 可能忽略了某些 $\{a\}$ 中的元素，而推論出「失真」的結果。這種資訊壓縮過程導致的保真度下降問題，也是必須考慮的重要議題。

6.9 發展階段

系統在發展的不同階段裡會進行不同的任務、發生不同的過程，本節當中，我們給出一個高層次的概覽，列出系統在「形成階段」與「語言階段」兩個維度的發展下新增的功能與發生的過程：

系統形成階段/語言階段	真值添加	系統運行與簡化
前語言	觀察真值 形成演序 觀察自身行動產生的演序 前綴演序分岔 形成定見（確定度高的抽象真值） 語言貼附	形成定形 資訊壓縮 －－－－－ 構造反身系統 －－－－－ 根據觀察到的演序選擇行動方案 依據可執行性與達成機率選擇行動方案 建立有效觀察真值的行動方案 －－－－－ 建立假說 建立驗證或否證假說的行動方案 －－－－－ 形成矛盾消解的後撤堆疊 找到否證時可增加系統穩定性的真值
語言使用	觀察以語言形式表達的真值與規則	使用語言表述推論鏈 創造詞彙表達複合概念 －－－－－ 使用語言影響其他理解語言的智能系統 對以語言形式呈現的真值表達表示否定 －－－－－ 經由使用語言的行動獲取真值

從上面的表格中我們可以看到，橫軸上，系統的「形成階段」是由真值添加走向系統運行、系統簡化。一開始，系統的表示域當中沒有太多的真值，透過觀察，系統當中的真值，包含規則等漸漸增加，並且開始透過抽象過程來使規則可以應用到更多的實體與情境上。同時，這時也開始發生「語言貼附」的過程，系統觀察到注意力域中重複同時出現的實體、概念以及對應的語言，漸次發展起語言使用的能力。

在前語言的「系統運行與簡化」階段，系統除了形成定形、構造反身系統（用自己在某些情況下將會採取某些行動來推測其它系統在那些情況下將會採取哪些行動，這可以透過構造具有遮罩的域來實現，關於遮罩，請見「策略與遊戲」一節）以外，也開始發揮行動與學習上的各種功能，這些功能不一定需要系統具備語言使用的能力，因此我們將它們歸在系統語言階段當中的「前語言階段」。

需要特別注意的是，我們在高階理論當中，仍然使用語言形式來表述與構造前述這些行動與學習功能，這是由於高階理論本身的「方法選擇」所致，高階理論是選擇使用了語言工具來表達與解析這些功能如何實現，但是這些功能在智能系統當中實際的實現途徑，並不一定需要系統具備語言使用能力，舉一個簡單的例子，我們日常觀察到的許多動物以及人類嬰幼兒也具備部分這些能力，但是他們可能並不具備或者還不具備語言使用能力。

在語言使用階段，系統可以觀察其所在環境當中「以語言形式」呈現的真值與規則，並以之作為系統真值增加的一個途徑。此階段下的「系統運行與簡化」功能比如使用語言表述自身想法、影響其它系統，透過與其它系統間的語言交互來獲取真值、創造語言當中的新詞彙與句法來表達觀念或構造長推論鏈等。

－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－

在系統的最佳化層面，目標是「增加有效的行動結果」:

	固定效益函數下的系統最佳化
增加有效行動數量 （固定行動方案組合、 可使用資源、 域真值）	調整定式組合 改變定式與其它方案間的優先順序 調整掃描順序 調整短期與長期目標域 減少無效行動 情緒調節 調整屏蔽閾值 調整行動域的啟動條件
最佳化行動方案組合 （改變行動方案組合）	增加行動方案數量 增加所採取行動方案的可行性（能夠成功執行） 增加所採取行動方案的確定性（成功執行時達成預期結果）

增加有效推論數量 （改變域真值）	增加推論鏈有效長度（避免斷鏈） 　　選取適當的真值組合進行記憶 　　形成複合規則 減少無效推論（避免無效鏈） 　　調整真值域的啟動條件 　　減少推論錯誤 　　減少真值錯誤

這裡，為了簡化分析，我們只討論系統在固定效益函數下的最佳化，而不討論系統效益函數改變的情形。

考慮系統內的「行動方案組合」、系統的「可使用資源」，與系統內的「域真值」，如果將這些因素都固定的話，系統就需要透過「增加有效行動數量」來達到「增加有效行動結果」的目標。這時，可以採取如表中所列的調整「定式組合」、「掃描順序」、減少「無效行動」等方式來增加給定時間內有效行動的數量，有關這些概念的討論，請見「行動」章節。

允許「改變行動方案組合」的情況下，系統的目標就會包含得出一個「最佳化的行動方案組合」，這包含增加行動方案的總數，以及透過排除不佳的行動方案或改變行動方案之間的優先順序，來試圖採用更有效的行動方案，增加實際採用方案的可行性與確定性。同樣的，這些概念的討論，請見「行動」章節。

由於推論是許多行動方案當中的步驟，且推論本身也是一種行動，因此「增加有效的行動結果」也包含增加「有效推論」的數量，這時常需要改變域中已存在的真值。在增加有效推論鏈的構造成功機率方面，我們可以篩除無效的真值、或者設置適當的鉤子、踏腳石等，來減少斷鏈機率，這在「推論距離」一節當中已經討論過。

另外，系統單位時間內可以進行的推論數量有限，因此我們也可以從「減少無效推論」的方向上下手。從這個方向上來說，除了調整「調整真值域的啟動條件」，避免構造出沒有效用的推論鏈以外，也可以透過減少推論錯誤（絕對錯誤）與真值錯誤（相對錯誤）來避免錯誤或無效的推論鏈產生。這些概念的討論，請見「錯誤」章節。

第七章：行動

7.1 效益函數

效益函數是系統用以評估某個狀態是否理想的方式，透過將狀態與一連串隱含「好」與「不好」的域內真值進行比對，就可以得到這個複合的狀態是「好」還是「不好」，以及「多好」或「多不好」。

比如，「餓肚子一個晚上」對一般人來說是「不好」，但是對於想要減肥的人來說，「餓肚子一個晚上，並且瘦下三公斤」很可能是「好」。「餓肚子一個晚上，並且贏得一棟豪宅」對許多人來說都是「好」，但是「不吃東西一百天，並且贏得一棟豪宅」很可能是「不好」，因為他們的系統中存在一個規則使得「不吃東西一百天」不是可接受的選項。

效益函數考慮的方向有「避免負面狀態」與「追求正面狀態」，而且對於大多數的智能系統來說，避免負面狀態更加優先。

避免負面狀態可以很簡單地理解為避免「不想要發生」的狀態，比如「餓肚子」、「流落街頭」、「受傷」、「被排擠」等等，對於一般人來說都是負面狀態。這些負面狀態之所以「負面」，是因為會影響到系統的存續性，或者與系統被植入的一些信念違背，比如透過說謊來賺錢，即使在不會觸法，與對方未來可能也不會有更多互動的情況下，對於某些人來說依舊會產生不適。

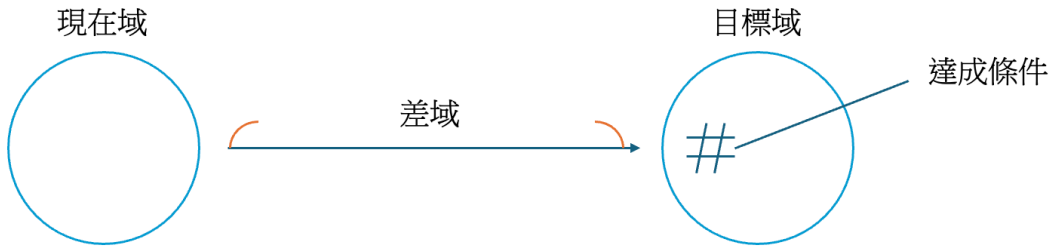
追求正面狀態可以理解為是追求「想要發生」的狀態，比如「賺大錢」、「被許多人喜歡」、「有良好的健康」等等，對於一般人來說都是正面狀態。這些正面狀態對於系統的存續性有幫助，有時候，如果並不是很直接地能夠看出是否有幫助的情況下，通常是能夠增加系統未來可選的選項，增加未來可選的選項時常是被正面認知的，所以「必定能從事，但也只能從事某個工作」即使在該工作待遇優渥、智能系統本身也喜歡這個工作時，也可能不會成為正面狀態。正面狀態也可能是與系統被植入的一些信念相符，比如「行善且不為人知」對於系統本身的存續性沒有幫助，但是對於許多系統而言，仍然是一個正面狀態。

一個系統的效益函數在任一給定時點，通常具有一定的模糊性，比如「願意花多少錢買一個特定的好看的背包」，一開始答案可能是一千元，那麼一千兩百元是否可以？或許可以。一千三百元是否可以？或許需要想一下。「被 A 討厭，但贏得 B 的喜歡」是否可以？或許需要想一下。如果 B 會非常喜歡你，是否可

以？或許可以。系統的效益函數如果不明確，就會影響到其行動決策，使得需要進行的推論量增加。

另外，系統的效益函數並不是固定不變的，會隨著其經驗發生改變，也可以透過系統內部的推論產生改變，這是系統最佳化的方向之一。更多有關效益函數的內容的討論，也可以參考經濟學相關書籍。

7.2 目標域、差域與達成條件



系統的行動源自於現在域與目標域之間的差域。

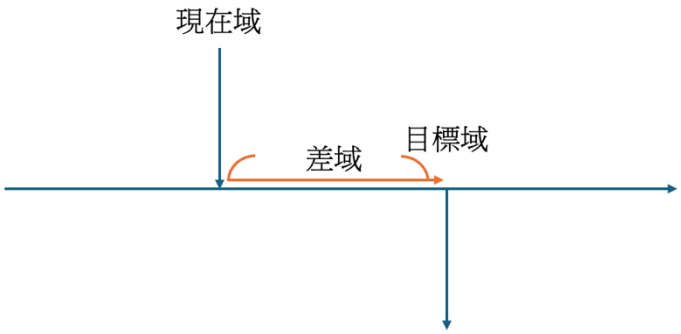
透過外界的資訊輸入，系統對於當下的事象會產生認知，形成現在事象域，然而，現在事象域在系統的效益函數評估下，幾乎總是並非最為理想的事象。

因此，系統會在現在事象域的基礎上，訂立出一個目標域，目標域的達成條件，也是其達成要件域，就是現在域當中的「負面狀態」對應的改善後結果。比如若現在域中有「肚子餓」，目標域的達成條件就是「肚子不餓」，現在域中有「無法買某個必需品」，達成條件就是「買到某個必需品」，現在域是「不被某個特定人喜歡」，達成條件就是「被某個特定人喜歡」。

目標域與現在域之間的差別，即兩個域之間的差域，有可能會隨著外部環境的自然變動實現，這時系統就不需執行行動來實現差域，比如現在域中有「下大雨」，而目標域達成條件是「不下雨」時，達成條件可能自然發生而不需系統執行行動，但是如果不會在系統不執行行動時，就透過自然變動實現，比如現在域中有「在戶外淋雨」，而達成條件是「在室內不淋到雨」時，系統就必須安排並執行行動以實現差域。

7.3 意圖

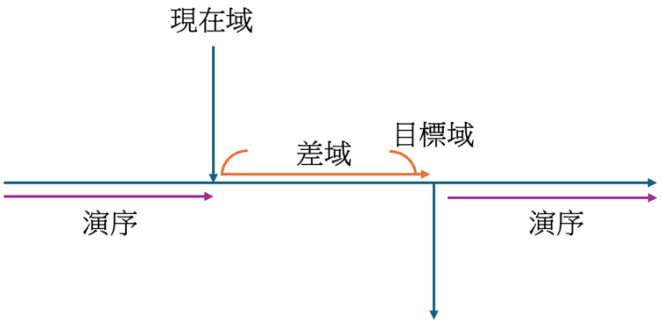
意圖是系統對於目標域的達成條件的描述。上一節中，我們已經提到系統在認知到現在域，且訂立了目標域後，會根據差域來安排行動。在高階理論中，我們以下圖來表示這個過程：



系統是透過輸入來認知到現在事象域，而系統的輸入習慣上畫在「上方」(見「系統」章節)，因此現在域處標示了一個向下箭頭指向中間的橫向箭頭。中間的橫向箭頭隱含時間的順序，左側是演序的上游，右側是演序的下游，目標域在橫向箭頭上，位於現在域的右側，且現在域在橫向箭頭上的位置，與目標域之間的一段就是對應到差域的演序。

大多數時候，目標域需要透過行動來達成，行動是系統的輸出，輸出習慣上在下方，因此目標域處，我們標示一個向下箭頭，表示目標域經由行動達成，這並非代表只有在目標域達成的這一個時點系統會執行動作，系統的動作實際上可能參與了對應到整個差域的演序。要指出的是，系統在現在「當下」的認知中，就已經設立了目標域，因此現在域到目標域的整個過程都是系統在現在域發生的這個當下進行處理與規劃的。

整體而言，在上圖當中，兩個縱向箭頭標示了系統從輸入到輸出的過程，橫向箭頭則是標示系統已經認知到，以及預測將發生的外部演序。



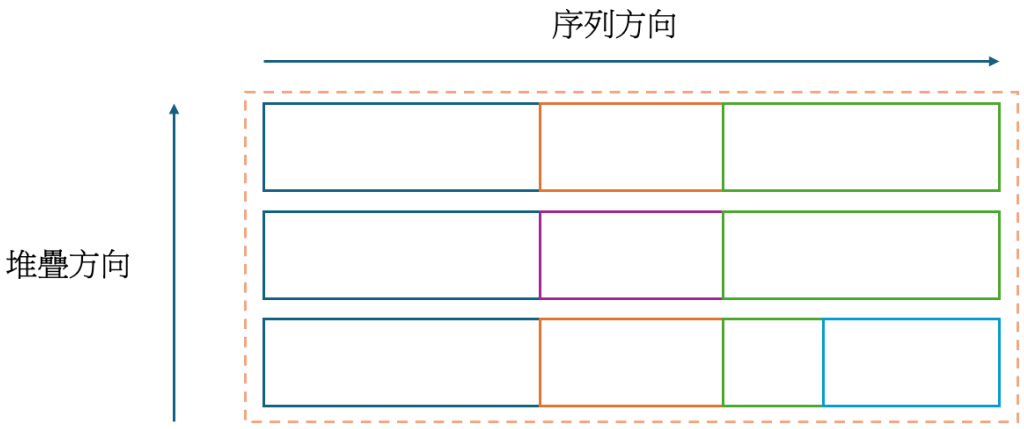
上圖當中，我們加上兩個與橫向箭頭平行的紫色箭頭來代表發生在現在域之前的演序，以及發生在目標域之後的演序。

系統在安排行動以實現對應到差域的演序時，必須考慮到在差域前與後的演序，這些演序之間隱含的規則不能衝突，因為系統要實現差域，必然是透過已學習到的、實現了過去的演序的規則（在考慮穿透變動，即「穿透產生的變化」的前提下）來找尋方法；這些使演序達到現在域，且透過演序達到目標域的規則，也會（在考慮了穿透變動之後）發生在目標域後的演序上。

因此，要實現差域，必須要使過去的演序、差域、目標域後的演序形成一個連續的序列。實現過去演序的規則經過穿透變動後，作用在目標差域上，且這些規則再經過穿透變動後，又將作用在目標域上，導致呈現出後續的變動。系統在安排實現意圖的行動時，必須要考慮差域前與後的演序，透過過去的演序來印證行動的確定性，且透過目標域後的演序來印證目標域在給定效益函數上的合理性。

7.4 行動序列與計畫堆疊

系統產生了意圖之後，需要透過安排行動來實現意圖。



上圖當中顯示了針對「單一個意圖」安排的行動。橘色框線中的每一個橫長方形，都是一個「行動方案」，或者稱為「行動計畫」。

每個行動方案，或稱行動計畫，都是由一系列的子序列構成，且每個子序列也都是由一系列的行動構成。在一個「計畫堆疊」當中，每個行動方案都是系統認為可以達成同一個意圖的不同方法。習慣上，我們將系統目前最優先執行的計畫放在最上方，往下是不同的替代方案。當因為新認知到的事象使得最上方

的計畫無法執行時，系統就會往下執行替代方案。

舉例而言，當系統的意圖是「喝水」時，行動方案可能包含「喝水壺裡的水」，且這個行動方案包含「從包包裡拿出水壺」、「打開水壺的蓋子」、「舉起水壺對準嘴巴」、「將水倒到嘴裡」等一系列的動作。然而，如果新的事象「水壺裡沒有水」導致這個行動方案沒有可行性，系統必須向下尋找其他替代方案，比如「去飲水機裝水來喝」。

「去飲水機裝水來喝」這個方案也包含一系列的動作，包括「找飲水機」、「倒水到杯子裡」、「舉起杯子對準嘴巴」、「將水倒入嘴裡」等等，每個子序列都有一個達成狀態，也就是子序列目標域的達成要件，比如「找飲水機」需要「找到飲水機」才算達成，「倒水倒杯子裡」需要「水進入到杯子裡」才算達成，當某個子序列的達成狀態無法實現時，系統可能決定往下方找尋能夠完成條件的子序列替代方案，或者在整個堆疊當中向下尋找另一個序列作為替代方案。

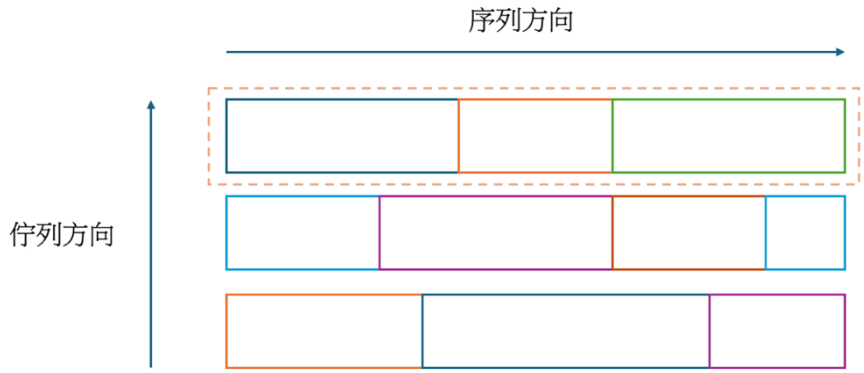
一個意圖對應的計畫堆疊當中，哪些行動方案在上，哪些行動方案在下，需要考慮方案的可行性與確定性。可行性指的是系統是否有辦法執行這個方案至每個子序列的達成要件都完成，這與系統認知到的事象、可支配的資源與推論能力有關，而確定性則指系統成功執行這個方案時，目標域達成的可能性，這與系統在預期行動序列中每個子序列會導致的演序時，使用規則的確定度有關。

並非所有規則的確定度（見「學習」章節中的介紹）都為一，有些規則是「很可能發生」，而非「必然發生」，比如「超市有賣瓶裝水」是很可能的，但是並非必然的，「超市有賣瓶裝水時，我有帶錢就可以買水」是很可能的，但也並非是必然的。系統對於規則的確定度的認知，來自於過去經驗到的演序的合併，而即使由系統過去經驗到的演序看起來，某個規則的確定度是一，系統通常仍然可以透過推論鏈構造出一些情形，使得規則確定度不為一。

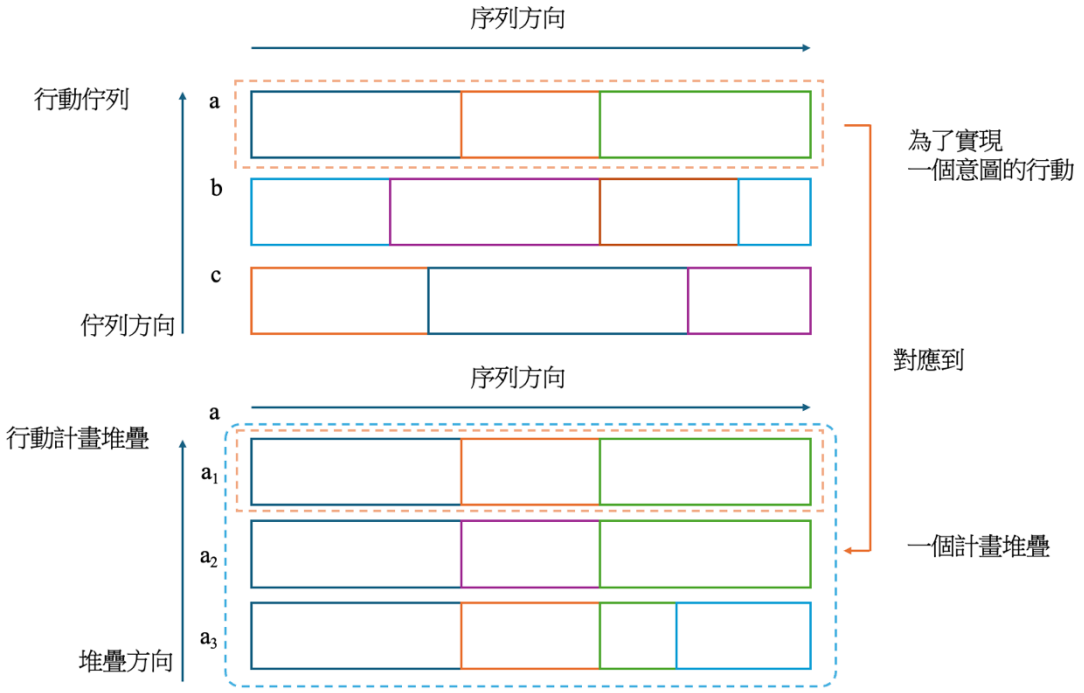
除了方案的可行性與確定性之外，系統也必須考慮方案將會使用到的資源，以及耗費的時間等，不過，因為系統的推論資源有限，多數情況下，都是直接使用計畫堆疊裡既有的方案，而不會額外評估是否有不在堆疊中的更佳方案，或者將既有方案重新排序。這是一個很明顯的路徑依賴情形，因為系統只是由於過去通常怎麼做，而且曾經達成過目標，所以這次也採用這個方法，而非重新透過組合不同行動對應的演序來尋找更佳方法。至於系統在給定的行動方案當中如何決定不同計畫之間的順序，則屬於系統最佳化的範疇。

7.5 行動佇列

「行動佇列」決定了接下來系統將透過行動實現的意圖，以及其他意圖之間的優先順序。



上圖顯示了一個行動佇列，橘色框線當中是為了達成一個意圖而安排的行動，對應了上一節中提到的一個計畫堆疊。



上圖中我們可以看到，行動佇列當中的每一個橫長方形，都對應到一個計畫堆疊，當一個意圖在佇列內部持續的順序變動當中排列到最上方時，系統就會關注到這個意圖，並且根據對應的計畫堆疊來安排行動。

－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－

意圖在行動佇列當中的位置會隨著時間變動，並不是固定等到最上方的意圖達成後，才將其移除，並且把下方所有的意圖按原順序上移，而是會隨時根據事象調整。系統在安排意圖的順序時，會考慮這些因素：

避免負面狀態：

- 避免危難
- 遵守規範
- 避免自身或其他系統不適

追求正面狀態：

- 追求自身或其他系統舒適
- 累積資源
- 改善系統

其中，避免負面狀態的因素之間排序較為固定，避免危難，即避免系統遇到傷害或重大不適的因素優先於遵守規範，遵守規範又通常優先於避免自身或其他系統不適，不過，不只「遵守規範」與「避免自身或其他系統不適」的優先順序並不一定，且與「不適」的程度有關，「避免自身不適」和「避免其他系統不適」之間的優先順序也不一定。越優先考慮的因素，就會使得對應的意圖被排在更前方，至於對於系統來說最佳的順序為何，則屬於系統最佳化的範疇。

追求正面狀態的因素之間排序則更為不固定，一般來說，追求正面狀態的因素都次於避免負面狀態的因素中的「避免危難」與「遵守規範」，不過，「追求自身舒適」、「追求其他系統舒適」、「累積系統可使用的資源」、「改善系統（即最佳化）」，與「避免自身或其他系統不適」之間，則沒有明確的順序，端看系統自身當下的效益函數，以及信念而定。

當佇列中的意圖的「目標域已經達成」，或者「新的事象使得達成目標域時的效益變動改變」時，意圖就會被移出佇列，比如「已經吃飽了」，就不需要追求「吃飽」，「氣溫升高了」，就不需要再透過穿更多衣服等方式追求「處於溫暖的環境」。

本質上，「目標域已經達成」和「新的事象使得達成目標域時的效益變動改變」都使得系統的效益不再能夠透過達成意圖而提升，這時，系統會轉而關注其他到達行動佇列最上端的意圖，並依此安排接下來的行動方案。

7.6 行動的決定

行動的決定開始於「狀態掃描」。

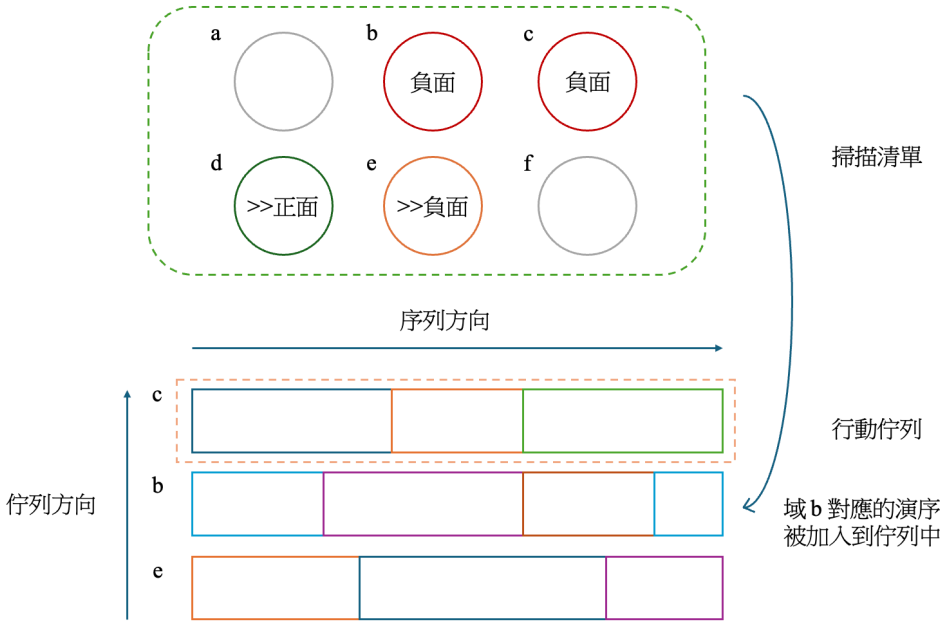
前面幾節當中，我們從「意圖的本質」、「單一意圖的實現」，討論到「意圖之間的優先度」。這節當中，我們從系統的輸入開始討論，依次探討輸入進入注意力域、意圖的產生、行動佇列更新，直到行動方案的決定。

當一個外部事件發生，且透過輸入域進入系統的認知當中時，如果新的狀態在注意力域裡，就可能會引發「事件廣播」。事件廣播發生後，使得「啟動域含有現在新的狀態」的一些域被啟動，進入被「掃描」的清單當中。

除了外部事件發生的情形以外，系統內部的推論也可能使得一些域被啟動，並且進入到掃描清單當中。

狀態掃描啟動時，系統重新檢視當前的掃描清單，並且針對清單中的域進行評估，並且根據域本身或即將推演出的演序，判別其是否將產生「負面狀態」或者「正面狀態」。

為了「避免現在的負面狀態」、「避免預測將發生的負面狀態」及「追求正面狀態」，系統會根據掃描清單當中現在或將會導致超過一定負面狀態「閾值」的項目訂立目標域，並將可實現差域的行動方案放入行動佇列當中。如果沒有超過負面狀態閾值的域，也可以轉為針對「追求正面狀態」訂立目標域，並將相關的行動方案放入行動佇列。

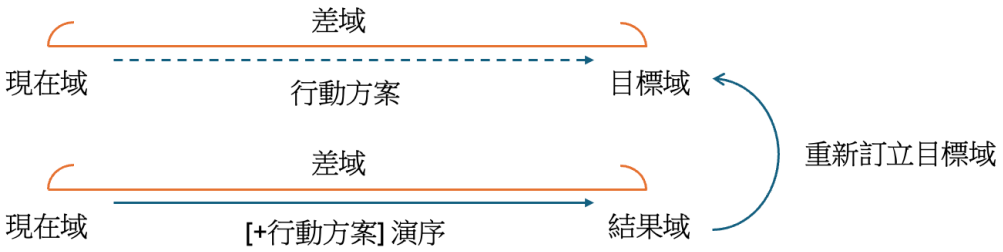


比如上圖當中，掃描清單中原先可能沒有 b，透過事件廣播，b 被增加到掃描清單，隨後，在系統進行狀態掃描時，因為要「避免現在的負面狀態」，b 對應的意圖被加入到行動佇列當中，它可能被加入到最上方，也可能被加入到較為下方，圖中顯示的是它被加入到原先已在佇列當中的 c 與 e 對應的意圖中間。

狀態掃描的發生是根據系統的「狀態更新頻率」，狀態頻率如果過高，掃描的過程可能不完全，如果過低，新的事件進入系統行動評估當中的所需時間可能過長。當系統正在使用推論資源以執行行動時，有時會需要進行「掃描屏蔽」，也就是避免新的狀態掃描佔用資源，以致降低當前行動執行的可行性。

－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－

透過掃描訂立目標狀態後，系統要尋找可以實現差域的行動方案，這個尋找過程可以透過逆推論鏈的方式進行。但是，在找到具備可行性的行動方案後，還必須要將現在域與行動方案結合，觀察其演序將會實現的結果域，如果結果域與目標域相同，代表這個行動方案具備確定性，在成功執行時可以達成目標，如果所有的行動方案都不能使結果域與目標域相同，就必須要根據結果域來重新訂立目標域。



重新訂立目標域後，因為目標域發生改變，使得差域同樣發生改變，這時針對新的差域，可能需要重新計算行動方案（因為雖然在前面的過程中，已經有一些行動方案可以達成結果域，但是或許有更好的行動方案同樣可以達成這個新的目標域，而沒有在前一次透過逆推論鏈生成）。重新計算行動方案後，同樣針對新的方案先評估可行性，再評估確定性，直到找到可以使當前（有可能經過數次更新）的目標域實現的行動方案。

－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－

綜上所述，可以將行動的決定簡化，表示為這樣的過程：

（1）目標域訂立階段（掃描形成意圖）

- 掃描清單更新
- 狀態掃描
- 訂立目標域

（2）行動佇列調整階段（更新行動佇列）

- 新的意圖進入行動佇列
- 行動佇列經過調整，更新關注的意圖

（3）行動決定階段（決定行動佇列前端的意圖如何達成）

- 根據目標域的達成狀態建立行動方案
- 根據可行性與確定性篩選行動方案
- 推測行動方案的執行結果
- （重新訂立目標域，並重複此階段直到推測執行結果與目標域相同）

－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－

我們可以把行動看作是一種思維（推論）過程，這是因為行動當中的身體控制、對話當中的表達選擇、決定當前執行中的行動方案是否仍然適用、判別某個步驟是否已經達成等，都需要透過推論來完成。因此，同樣行動同樣也會佔用到系統的推論帶寬。

另一個方向上，我們也可以把系統進行的推論工作看作是一種行動，與其它行動一同參與行動佇列上的排序，並在到達佇列上端時被執行。這裡，可以將推論分成兩種，一種是「與進行中未完成的行動有關的推論」，一種是「與已完成或未開始的行動有關的推論」。

與進行中未完成的行動有關的推論，比如評估當前行動方案是否仍然適用、進行到哪個步驟等，是行動方案當中的一部份，會在行動方案到達行動佇列的上端後，在執行該方案的過程當中發生。

而與已完成或未開始的行動有關的推論，比如反思已採取某個行動方案完成的行動是否有更好的方案、壓縮觀察得到的真值、對尚未開始的行動進行規劃等，本身在一個含有許多推論任務的推論佇列（裡面也包含與行動無關的推論）裡排序，這個推論佇列作為行動佇列中的一個任務，會跟著其他任務（意圖）一起排序，但是內容都完成時，不會被排除，而是回到行動佇列下端。

如果行動佇列當中，當前其他需要執行的行動都已完成，而推論佇列到達行動佇列的最上端時，這時系統就按照推論佇列當中的推論任務順序，同樣由最上端的推論任務開始處理。比如，如果現在是睡前，沒有其它行動需要被執行，系統可能就開始重新回想當天發生的事件，思考已執行的行動是否有更好的作法等等，完成一個推論任務，或者用盡給予該任務的時間後，就繼續往下處理其它推論任務，比如開始安排明天要舉行的某個會議的流程。

7.7 行動定式與行動定形

行動定式是一系列的動作，通常可以實現特定的差域；從開始域經由行動定式而產生的一連串演序與執行的行動，結合起來就畫出行動定形，也就是說，行動定式如預期順利進行時，會將行動定形一路從開始域完整畫到完成域。

行動定式是系統為了「節省推論資源」與「應對模糊環境」而根據經驗所訂立的，由於本質上，行動定式與定形都會具有路徑依賴的缺失，所以在推論資源沒有限制時，如果還是根據定式執行行動，總是只能達到差於或等於最佳化的結果，是一種「受限最佳化」的例子。

當定形的開始域啟動條件被滿足時，對應的定式會被放到佇列當中，並且當定式到達佇列的最上端時，系統就會執行定式的內容。與一般的行動安排不同的是，系統在執行定式的內容時，不一定是在訂立了目標域的情況下，為了實現差域而執行，也不一定會將定式構成的行動方案與現在域結合而觀察結果域是否理想，因此，許多時候，系統會在執行的過程中，或者完成後，才意識到不應執行定式。

比如，有時候我們會發生「才剛刷了一次牙，又刷了一次牙」、「明明戴著眼鏡，卻在尋找眼鏡」、「已經知道常走的路線在施工不能通行，卻還是走到該路線」等等的情形。如果在執行動作方案前，經過完整的訂立目標域、評估行動可行性與確定性的過程，就不會發生這些情形，但是為了節省推論資源，系統在某個行動定形開始域的啟動條件被滿足時，就把對應的定式安排到佇列當中，使得系統進行了多餘或錯誤的動作。

雖然有這樣的缺點，但因為節省推論資源的好處更大，或者不確定性很高（先吃了早餐才出門，卻發現目的地有很多好吃的食物，但這可能無法事先得知），所以定形與定式仍然是系統安排行動時的重要的機制之一。

這裡列出一些行動定式，讀者可以思考一下它們各自的開始域啟動條件：

- 上廁所、刷牙、洗臉
- 擦乾身體、吹頭髮
- 看燈號、看兩側來車、穿越馬路
- 搭上公車、感應票卡或投幣、找位子坐下
- 拿起水壺、打開水壺的蓋子、喝水、關上水壺的蓋子

7.8 對話

對話是系統進行複雜行動決策的一個例子。對話與來自其他系統的輸入非常相關，而且也大量仰賴系統對於其他系統狀態的推論與預測。

進行對話時，輸入是其他系統的語言表達，可能以聲音或文字形式呈現（文字形式比如使用打字對話）。當接收到來自其他系統的表達輸入時，系統將其視為一個事件，並且進行事件廣播，以更新系統內相關的域的狀態。

這時，一些域的啟動條件被滿足，因此進入掃描清單當中。系統此時需要決定是否對於其他系統的表達進行回應，無論是為了遵守規範，或者避免其他系統或自身的不適，或者追求其他系統的舒適，此時系統可能會決定回應對方。當然，有時候在考量了負面與正面的所有因素之後，系統也可能決定不予回應。

這裡，為了表達清晰，我們將關注的系統稱為系統 a，它的對話對象稱為系統 b。當系統 a 決定進行回應系統 b 時，首先必須決定目標域。從系統 b 的表達，以及整個對話的先前內容當中，系統 a 應該可以對於系統 b 的狀態有所判斷，比如當 b 問出「你吃飽了沒？」時，a 可能判斷 b 並沒有「a 已經吃飽」或「a 沒有吃飽」的真值。同樣的提問，根據對話的先前內容與當下 a 的狀態，也可能是解讀為 b 認為「a 已經吃太多，不應該再吃了」。

如果系統 a 判斷系統 b 的狀態屬於並沒有「a 已經吃飽」或「a 沒有吃飽」的真值，因此提問的意圖是屬於「獲取其他系統真值」時，a 需要決定如何回應。從過去經驗學習而來的各種回應方式當中，a 可能有這些回應方式：

- 已經吃飽了
- 還沒吃飽
- 還沒吃飽啊
- 我還沒吃

- 我有吃過了
- 跟你有什麼關係？
- 怎麼可能已經吃飽了？
- 你猜猜看
- 不是跟你一起吃的嗎？

除了這些回應方式以外，還有其他各式各樣的回應方式，但是並非無限多種。這些回應方式之間，有一些與其它某一些的意義相同，有一些差別在語助詞，有一些是對應系統認知到的、與本身有關的真值，有一些則是對應系統認知到的、與對方系統有關的真值。

比方說，同樣是認為 b 不知道自己是否真的有吃飽，而且 a 實際上沒有吃飽，a 可能回應「還沒吃飽」，也可能因為與 b 之間的關係疏遠而選擇反問「跟你有什麼關係？」，或者，可能因為時間太早而反問「怎麼可能已經吃飽了？」，或者為了玩笑、邀約、反諷等原因，回答「你猜猜看」，另外，也可能因為不想要 a 發出邀約，或者不想繼續此話題，因此回答與真值相反的「我吃過了」。

這些不同的回應方法之間意義不同者，對於 a 而言都對應到不同的現在域與目標域的組合。根據 a 所認知到的現在域以及目標域，a 可以從所知的回答方法裡面選擇符合現在域以及所期望的目標域組合的方式來回應。這些不同的回答方法是 a 從經驗當中習得的。

當 a 無法理解一種回應方式對應的現在域及目標域時，一般來說，它就不會去使用這種回答方式，除非 a 仍然在語言學習的模仿階段，或者它為了特定的理由，比如要造成對方的混亂，或者停止對話，而刻意採用了「不合常理」的回答方式。比方說，a 可能使用「西瓜是綠的，香蕉是黃的」來回答 b 的提問「你吃飽了沒？」，此時 a 的意圖不是繼續對話，而 b 因為無法理解這個回應的意涵，可能追問「什麼意思？」或者停止對話。

在選擇回應方式時，a 可能將所有經驗過與可推論而出的合理回答方式列舉出來，並且把每個回答方式（行動方案）與現在域結合，來推估實現的結果域，接著，再根據每個結果域的效益函數來決定採用哪一個方案。

然而，在實際的對話當中，受限於有限的推論能力（回應的延遲很可能不能過長，否則將會很快造成對方系統的不適，這在多數時候是系統傾向避免的），系統必須要以低推論的方式來進行回應選擇。這時，系統會將對方的表達與相關

真值結合在一起，並且尋找啟動要件被這樣的真值組合覆蓋的域，也就是說，系統不是先訂立出一個理想的目標域，也並非將所有的行動方案列出後計算結果域之間的優劣，而是仰賴與對話有關的行動定形來選擇回應方式。

啟動要件被當前的真值組合覆蓋的域，有可能位在某些行動定形的某些節點上，這時候，系統去檢視在這些定形上，是否有較理想的下游的節點（檢視的完整程度受限於可使用的回應時間），並且採取定形當中可以往該節點移動的行動。由於「定形」是那些常常重複畫出的圖形，這代表沒有充分時間思考時，系統會採取那些過去「自身常使用」或「常聽到別人使用」的回答方式。

這種做法當中，雖然系統可能還是有考慮過超過一種回應方式，但是遠非全部的可能選項，系統只能針對有考慮到的幾種回應方式，根據對應的定形檢視可能實現的目標域，並在有限的推論資源與合理的回應時間當中，選定一種方式回答。這也告訴了我們，因為具有強烈的路徑依賴特性，多數時候智能系統的回應都是「受限最佳」的，系統在一般的對話當中，幾乎總是無法進行完善的計算（這裡，經過完善的計算是指系統考量過自身知道的所有可能使用的回應，以及這些所有回應結合目前的事象域後，根據自身經驗後續可能導致的情形）後才選擇回應方式。

從上面的內容當中，我們可以看出系統在實現對話功能時，有兩個階段：

（1）學習階段

- 觀察其他系統之間的對話，以及相關真值與雙方系統的反應
- 透過對話、相關真值與雙方反應（演序）來構造自身的對話系統
- 在接收到類似的表達時，模仿已知的回答方式來印證演序

（2-1）執行階段 － 最佳化模式

- 判斷對方表達、相關真值，以及對方系統的狀態
- 列舉出所有經驗過與可推論而出的合理回答方式
- 針對每種可選的回應方式，計算結果域
- 透過一系列結果域在效益函數上的表現，選擇回應方式

（2-2）執行階段 － 快速回應模式

- 判斷對方表達、相關真值，以及對方系統的狀態
- 找尋含有符合當前情況的節點的行動定形
- 針對找到的定形，檢視可能發生的下游節點是否理想（在效益函數的

計算下，超過一定的可接受門檻）
- 選擇定形上會往該下游節點方向移動的回應方式

另外，系統也可能不是回應對話當中對方的表達，而是主動開啟話題。這種開啟話題的情形與有充分時間選擇回應方式的情形比較類似，系統可以試圖採用最佳化模式來實現某些意圖。常見的意圖如下：

- 獲取外部真值
- 描述內部真值使其他系統獲知
- 影響其他系統的行為
- 影響其他系統認知的真值
- 影響其他系統的情緒

7.9 表達與寫作

使用寫作或長篇演講的形式來表達時，系統必須預測整個過程當中資訊接收者的表示域的演變。在寫作時，必須考慮「若不先講哪些主題，讀者無法理解這個主題」、「看完了前面的段落後，讀者已經知道某些資訊」、「由於前面的表達方式，讀者現在很可能是某種情緒」等等，與資訊接收者在接收資訊到某個時點時的狀態有關的因素。

整體來說，寫作（或演說）通常有這些目的：

- 描述內部真值使其他系統獲知
- 影響其他系統的行為
- 影響其他系統認知的真值
- 影響其他系統的情緒

除了這些透過對話亦可以達成的目的之外，由於寫作的留存特性，也可以透過寫作來克服系統本身的有限記憶與推論距離限制。

具體而言，寫作使得系統可以將可能不會進入長期記憶而導致丟失的資訊，透過寫作留存，使得系統本身在後續時點仍然可以取得這些資訊，另外，因為長推論鏈要求記憶的真值數量可能超過系統的工作記憶，寫作可以讓系統把推論鏈的中間結果留存下來，以便實現更長的推論，也就是實現我們提過的「踏腳石」與「扶手」的作用。

系統在實現寫作功能時，有兩個階段：

（1）學習階段

- 觀察其他系統對於某些真值域的描述方式，以及自身接收後的演變
- 透過其他系統的表達對於真值域的貼附，與自身演變構造寫作系統
- 在需要進行類似的表達時，模仿已知的表達方式並印證對演序的預測

（2）執行階段

- 判斷寫作的目的，如描述真值、影響其他系統行為或情緒、記錄等
- 描述真值或進行記錄時，先找到被要描述的真值域覆蓋啟動條件的域，使用貼附於這個域的表達方式進行表達；接著，計算餘域，再對於餘域以同樣的過程進行表達
- 影響其他系統行為或情緒時，找到經由經驗或推論得到的，具有這樣的效果的表達方式，在代換必要的真值後進行表達

我們會發現，由於「尋找被要描述的真值域覆蓋啟動條件的域」，並且接著再「尋找被餘域覆蓋啟動條件的域」的這個過程具有強烈的非唯一性，因此系統既不會總是用同一種方法來進行較長的表達，在寫作或演講完成後，也幾乎無法在不閱讀紀錄的情況下用同樣的字句完全將其復現。

由於寫作的過程屬於非常複雜的行動決策，而且對象不固定，因此需要進行的推論遠超過系統可以合理負擔的數量。系統因而會強烈仰賴行動定形來完成寫作，而行動定形在達成某個目標的同時（比如描述真值），也可能無法同時達成其他目標，甚至造成負面狀態（比如在沒有這樣的意圖下，使其他系統產生負面情緒），對於行動定形的仰賴程度，以及系統進行搜尋的行動定形組合本身的優劣、不使用行動定形時，行動方案的多寡與確定度等，都會影響到寫作的品質。因此，寫作是同樣作為智能系統的不同個人，在表現上會差異很大的任務類型之一。

同時，無論是對話或是寫作，都會遇到向其他系統描述外部或系統內部真值時，與對方的系統內部已經存在的真值（或真值隱含的推論）矛盾的情形，此時，因為對方的系統可能選擇接受（透過前綴分岔或覆蓋原本的真值）或不接受，所以如何使用「有說服力」的方法來使接收資訊的系統更傾向於選擇接受真值，也是進行表達的系統的考量之一。一般來說，這可以透過學習階段觀察其他系統對於不同真值域的描述，以及當這些描述與正在進行學習的系統本身

已經存在的真值矛盾時，何時系統更傾向於接受而非排斥這些真值來習得。

至於系統接收外部資訊而產生矛盾時內部如何進行解決，更多相關討論請見「學習」章節。

7.10 輸出譜

系統具有有限數量的輸出鍵。透過觀察以不同組合與順序啟動輸出鍵形成的演序，系統可以學會如何控制其「身體」，並進而影響其他環境中的物體。

系統對於輸出譜的掌握程度會影響到行動方案的可行性，事實上，我們可將行動方案的可行性分為兩部分，其一是訂立方案後，對輸出譜的控制與預期的身體狀態相同的機率，另一則是與預期的身體狀態相同或不同時，可以達到預期中的系統外部狀態（不含系統的身體）改變的機率。

由於行動執行的品質，即使「滿足目標域達成條件」，仍可能因為花費時長、細微的量、位置、角度不同等不在系統認知內的因素而影響到後續的演序（比如站到桌子上換電燈泡，雖然有換好，但是差點跌倒，或者桌子壞掉）。雖然對於系統來說目標域仍然達成，但是後續的演序與預期中不同，因此在演序合併的過程當中，系統可能會選擇調整目標域，以使該行動方案不再被應用在同一個意圖上，或者，也可能在支持該目標域的證據強烈時，選擇透過增強對於輸出譜的掌握來提升行動執行的品質。

有關輸出譜的具體形式與設計，屬於生物學或機器人學的範疇；有關系統透過學習提高對輸出譜的掌握，也可使用機器學習等低階方法；有關系統在觀察到演序與預測中不同時進行內部調整，請見「學習」章節中的討論。

7.11 策略與遊戲

遊戲的核心是「遮罩域」，當中會描述「好」或「不好」的條件，而形成遊戲中的效益函數，另外，也可能會包含有關指定環境及與可選行動組合的真值。

a 域經過 b 域「遮罩」後所產生的域 b(a) 中，基底是所有 b 的基底，加上所有不與 b 的基底衝突的 a 的基底。b(a) 中含有所有 b 中的真值，而當含有 a 中的真值 A 的域 {A} 與 b 的和域 {A}+b 不會在 b 中額外增加矛盾（b 中原本可能也有矛盾）時，A 也在 b(a) 中，反之，則不在 b(a) 中。

我們也可以將遮罩 b(a) 看作是以 b 作為外部域，以 a 作為內部域，並且當不會增加外部域中的矛盾時，也可以允許 a 中的真值向外穿透。

a 域經過 b 域「遮罩」，代表「有衝突時，都以 b 域優先」，屬於一種「覆寫」的概念，在「七個子連成一線才贏的五子棋中」，七個子連成一線才贏的規則將原本五子棋的域覆寫，透過真值遮罩改變了原本域內的內容。事實上，原版的五子棋規則「先把五個子連成一線就贏」的規則，也在參與遊戲的系統中的域上發生遮罩，因為當不考慮五子棋這個遊戲本身時，「先把五個子連成一線就贏」的真值通常是不存在的。

智能系統可以根據其效益函數來決定參與哪些遊戲，而遊戲的定義可以使系統的任何意圖都被納入在一些遊戲當中。比如，我們可以把「準時到辦公室」看作是一個遊戲，也可以把「選出一個旅遊目的地」、「規劃交通路線與工具」、「訂票」、「向櫃檯人員詢問到所需資訊」都看作是遊戲。當我們以遊戲的角度來看待這些系統域達成的任務時，目標域具有特定的真值，比如「向櫃檯人員詢問到所需資訊」需要達到「知道所需資訊」（櫃檯人員講了而系統沒有聽到不算達成遊戲目標；反之，不經詢問，櫃檯人員或其它人就講出所需資訊，卻可以算是達成遊戲目標），「訂票」的目標域中有「訂到票」，「規劃交通路線與工具」的目標域中有「已經規劃出可行的交通路線與工具」等。

有些遊戲直接與身體相關，比如「將衛生紙團丟進垃圾桶裡」、「摸到天花板」、「把一隻狗抱起來」，這樣的遊戲與系統對身體的控制能力有緊密關聯。

有些遊戲則落在我們一般認知的「遊戲」範疇內，比如「21 點」、「比大小」、「圍棋」、「足球」、「猜拳」等等，這類遊戲都有清楚的一系列真值形成的「遮罩域」，當智能系統參與在這樣的遊戲當中時，這些遮罩域中的真值規範出的「贏」或「輸」的條件會覆寫參與者本身的系統中真值，從而改變系統所欲達到的目標域，並進一步影響到其行動計畫。比如，一個系統可能特別喜歡「數字 1」，特別不喜歡「數字 6」，若非參與比大小的遊戲，它會想在擲骰子時擲出數字 1，不想擲出數字 6，然而在「比大」的骰子遊戲當中，這個系統所制定的目標域受到遮罩真值影響（前提當然是系統有「想贏」的效益函數因子），進而改變目標想要擲出數字 6，假如這是一個聲控骰子，喊出什麼數字就會擲出什麼數字，那麼系統可能就會因此制定喊出數字 6 的行動計畫。

在許多遊戲內，行動的選項是受限的，形成一個參與的智能系統可選的行動組合。比如，有些遊戲會使得參與的系統當下只能下黑子、只能下白子、只能下

在線的交界處、只能下在框出的格子內、只能用腳踢球、只能問是或否、...這是透過遮罩真值來直接影響系統針對當前目標域所制定出的行動（犯規時，可能會輸，使得目標域無法達成）。如果遊戲的特性使得不按照可選的行動組合來安排行動，也不會有明顯的負面效果，那麼可能就會有一些系統使用有創意、特立獨行的方式來參與遊戲，而如果遊戲對於行動組合要求嚴格，使得不依所述的行動組合選擇行動時會有強烈負面效果，則參與的系統會更加有動機從中選擇並安排行動。

並非所有參與在遊戲當中的系統都能夠選擇執行受限行動組合中的所有選項，比如受到身體限制，可能無法按出複雜的按鈕組合、無法將手伸到特定高度等等，不同系統在同一個遊戲當中，面對的可執行行動組合可能並不相同。

－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－

討論「遊戲」的目的是為了簡化對於系統的行動決策的理解或設計過程。遊戲之間可以依據不同的特性分成不同的類型，有些是「達成或未達成」，比如「搭上某班列車」，有些是「單一因子最佳化」，比如「把球丟得越高越好」，有的是「雙因子」或「多因子」最佳化，比如「找到一家便宜又好吃的餐廳」，有的是混合類型，比如「花最少錢買到電視」是「單一因子最佳化」與「達成或未達成」類型的合成，因為盡量花較少錢是其中的最佳化部分，但是如果因此沒有買到電視，在遊戲形成的效益函數之下，一樣無法達到理想的目標域。

智能系統在面臨新的遊戲時，通常並不是從頭開始透過自身認知域中的真值來經過大量計算擬定出策略，而是試圖選取一些已存在系統中的、經驗上適用於某些遊戲的範式，並優先組合這些範式當中的定形、定式等，來應用在當前面對的遊戲上，而其中的實體、規則等，會需要根據遊戲的遮罩域進行調整，使得範式可以成為當前遊戲中可用的指導原則，這個過程稱為「範式複用」。

當多個複用在同一個遊戲上的範式間發生衝突時，需要依據當前遊戲的規則、範式中規則的確定度等來決定如何消解矛盾，經過融合的範式會成為系統應對當前遊戲的初始「策略」，並且在其上繼續透過假說設定與驗證、觀察行動結果等過程累積新的真值，使策略更加完整，當策略完善以後，這個新的策略也會成為系統未來可用在別的遊戲上的一或多個範式。

綜上所述，我們可將智能系統遇到的任務與展現出的功能以「遊戲」層級來定義與檢視，這些遊戲具有經規範的效益因子，會與參與系統本身的效益因子合成，成為系統參與時的效益函數，並據此形成系統在遊戲中的目標域。

系統在制定遊戲中的行動策略時，也必須考慮遊戲遮罩域中對於可選行動方案的限制。智能系統會根據遊戲的特性，選取經驗中適用於相同或類似遊戲上的範式，經過範式複用與融合的過程，形成對於遊戲的「策略」，並透過參與遊戲的過程，進一步使「策略」更加完善，並成為未來複用於其他遊戲的基礎。

第八章：複雜推論

預印本無此章節內容

第九章：學習

9.1 學習的過程

學習過程描述了智能系統的功能是如何改變的。

學習經由兩種過程實現，一種是系統表示域中真值的增加，另一種則是域中的矛盾消解。我們可以將前者稱為「初次學習」，後者稱為「二次學習」。

當智能系統在環境中透過輸入認知到的事實，不僅與表示域中已存在的真值不衝突，也不為表示域添加新的基底時，我們將它稱為「不添加真值」。

當智能系統在環境中透過輸入認知到的事實，既與表示域中已存在的真值不衝突，且為表示域添加了新的基底時，我們則將它稱為「添加真值」。

「添加真值」與「不添加真值」都屬於「不衝突真值」，其中表示域裡增加了「添加真值」的過程，屬於「初次學習」。

「不添加真值」的來源除了系統在環境中透過輸入認知到的事實以外，由於智能系統不完全的特性，使得表示域通常處於非緻密狀態。在這樣的「半密域」當中，僅透過既有真值而推論出原先不在域中的真值，也屬於「不添加真值」增加的一種方式。

至於智能系統在環境中透過輸入認知到的事實，與表示域中已存在的真值衝突時，無論是否同時添加新的基底（我們可將衝突的部分與不衝突僅添加的部分分開檢視），都稱為「衝突真值」。也可以這樣來看：當「域內無衝突的 a，與同樣域內無衝突的 b 之間的和域 c_1 域」內沒有矛盾，而「a 與 b+{A} 的和域 c_2 域」中有矛盾時，A 對於 a 域而言是衝突真值。

比如下面的這個例子：

{a}{～C}

{b}{B, R_1=A^B->C}

{c_1}={a}+{b}={～C, B, R_1}

{b}+{A}={A, B, R_1}

({c_2}={a}+({b}+{A})={～C, A, B, R_1}) [R_1]-> {～C, A, B, R_1, C}

系統的表示域中增加衝突真值的過程，我們稱為「二次學習」。

－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－

當系統的不衝突真值是由觀察外部環境而得時，我們稱之為「外生初次學習」，僅由內部推論而得時，稱之為「內生初次學習」。

類似的，衝突真值由觀察外部環境而得時，我們稱之為「外生二次學習」，僅由內部推論而得時（內部推論出的規則與已存在的真值矛盾），稱之為「內生二次學習」。

綜合來看，系統真值的增加有這些來源：

（1）外生來源

- 由感官接收非文字資訊
- 由感官接收文字資訊
- 觀察到演序

（2）內生來源

- 半密域中的新推論結果
- 規則合併
- 演序合併

（3）假設

- 經推論定式產生的，未被否證的假說

由於規則合併與演序合併也會產生新的規則，而規則同樣是真值，因此屬於真值增加的途徑。

除此之外，系統內除了觀察到或推論出的真值以外，還有一類「仍待決定是否有真值」的描述，也就是系統中的「假設」或稱「假說」。我們可以使用與一般真值同樣的表述方式與規則來操作這樣的真值，不過由於其並非系統已確定的真值，而是用以進行抽象思考與預測的工具，所以我們將這類真值透過添加前綴，放在表示域內的「假設域」中。

更多關於假說的討論，請見「假說與為了學習的行動」一節。

－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－

系統在接觸到新的「遊戲」時，經過瞭解規則、整合真值、複用範式、驗證假說的一系列過程來擬定策略。擬定初步策略後，在行動的過程中，又會認知到新的真值而使範式有所調整，以及重新訂立與驗證假說。針對遊戲的策略擬定過程本質上是一種多輪的學習。

初次擬定策略

- 初次學習規則（透過觀察或其他系統的解說）
- 針對所學習到的規則，在一定推論距離內導出相關的真值集合
- 尋找可複用的範式以減輕推論負擔
- 根據效益函數擬定目標域
- 根據目標域選擇複用的範式
- 形成假說
- 透過嘗試行動或詢問其他系統，驗證或否證假說
- 將規則導出的真值集合、經過驗證的假說與複用範式結合，形成策略
- 根據策略安排遊戲中的行動

修正與改善策略

- 根據安排的行動計劃執行行動
- 經由觀察到的行動結果或其他系統的解說二次學習規則（真值改變）
- 重新尋找可複用的範式
- 根據效益函數，重新擬定目標域
- 根據新目標域，重新選擇複用的範式
- 形成並驗證或否證新的假說
- 形成新的策略
- 根據新策略，繼續安排遊戲中的行動

我們可以將智能系統的「學習」視為其針對所有參與的遊戲擬定策略的過程，這些遊戲的類型包含預測環境變化、得知環境真值，及達成特定目標域等。

「得知環境真值」與「預測環境變化」，都可以讓系統後續擬定更多有效的策略以「達成特定目標域」。給定智能系統的初始狀態，經由多次擬定策略並執行行動以「達成特定目標域」的過程，系統可以試圖使自身更加接近「對自身而言效益更高」的狀態，這也是智能系統學習與行動的目的。

9.2 三角勾稽

當表示域中的推論規則存在前提皆為真，而結論不為真的情況時，我們就稱域中產生了「矛盾」。矛盾會使域的穩定度下降，也使系統因為結構中產生不必要的複雜部分而效能降低，要排除系統中的矛盾，需要進行矛盾消解。

矛盾消解處理的狀況可以統一表示為兩類情形，其一為：

$$\{a\}\ \{R_1=A \text{ -> } B, A, C=\sim B\}$$

我們將形同 $\{ R_1=A \text{ -> } B, A, \sim B \}$ 的域中真值情形稱為 R_1、A、～B 的「靜三角」。上例中，即出現了 R_1、A、～B 的「靜三角」矛盾。

另外一種情形為：

$$\{b\}\ \{R_2=\{\| t_1\}\{A\} >>\{\| t_2>t_1\}\{\sim B\}, \{\| t_1\}\{A\}, \{\| t_2\}\{B\}\}$$

我們將形同 $\{ R_2=\{\| t_1\}\{A\} >>\{\| t_2>t_1\}\{\sim B\}, \{\| t_1\}\{A\}, \{\| t_2\}\{B\} \}$ 的域中真值情形稱為 R_2、A、B 的「動三角」。上例中，即出現了 R_2、A、B 的「動三角」矛盾。

－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－

靜三角矛盾可以用式 1 表示（當然，$\{R_1=A \text{ ->} \sim B, A, B\}$ 也可以）：

$$\{a\}\ \{R_1=A \text{ -> } B, A, C=\sim B\}$$

式 1

式 1 中，矛盾消解的方法有這三種：

- 假設非 {a}{A}
- 假設非 {a}{C}，即非 {a}{～B}
- 假設非 $\{a\}\{R_1\}$，即非 {a}{A -> B}

至於使用三種方法中的哪一種，在無錯誤系統中，可以考慮真值 A、C、R_1 的確定度何者較低，或者 {a} – {A}、{a} – {C}、$\{a\} – \{R_1\}$ 何者的「穩定度」較高。

比如，真值 A 的確定度最低時，就假設非 {a}{A}，如果真值 A、C、R_1 的確定度相當時，再考慮 {a} – {A}、{a} – {C}、$\{a\} – \{R_1\}$ 何者的穩定度最高，比如當 {a} – {A} 的穩定度最高時，假設非 {a}{A}，以此類推。

當 {a} – {A}、{a} – {C}，與 {a} – {R_1} 的穩定度相當時，可以再視 A、C、R_1 何者在域中對應基底元素的「權重」較低而定，當 A 的權重最低時，假設非 {a}{A}。

關於「確定度」、「穩定度」與「真值權重」的討論，請見「確定度、穩定域與不穩定域」一節。

假設非 {a}{R_1}，不一定代表要將 R_1 從 a 當中完全移除，一種消解的方式是在 R_1 上增加前綴，比如 {a}{{|| t_1}{R_1}}，比如將規則「沒有颱風的時候要上課」變成「平日沒有颱風的時候要上課」。

－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－

在上述的三種矛盾消解方法當中，非 {a}{A} 與非 {a}{C} 是把 A 與 C 自 a 中移除，來避免矛盾發生，這是因為我們假定 A 是 R_1 的「合法前提」，使得根據 R_1 規則，域中必定可推導出 B，而與 C=～B 產生矛盾。然而，當 R_1 是抽象規則時，還有一種可能是 A 並非 R_1 的「合法前提」。

比如，小美可能和小明說「我在家裡養的『小晴』是顏色鮮豔的杜鵑」。這時，小明本身的表示域中有以下真值：{

A.=.<小晴 ||i 鳥>.被.養.在.家.裡

R_1.=.1/鳥.被.養.在.家.裡 -> 需要.準備.空間.讓.1/鳥.活動

B.=.需要.準備.空間.讓. <小晴>.活動

}

但是，小明和小美說「你住的地方太小，不適合小晴」時，小美可能和小明說「怎麼會，小晴又不需要空間」。

假設「A=<小晴>.被.養.在.家.裡」和「C=不.需要.準備.空間.讓.<小晴>.活動=～B」都是真的，R_1 可能不是真的，然而，A 也可能不是 R_1 的合法前提，因為小美所提到的「杜鵑」可能不是鳥，而是「杜鵑花」，因此可能出現 A、R_1、C 都有真值的情形。這種可能性屬於系統「錯誤」中的「擴張錯誤」，因為抽象規則透過擴張套用到真值上時，出現了套用錯誤而產生錯誤推論的情形。

－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－

動三角矛盾可以用式 2 表示：

{b} {

R_2={|| t_1}{A} >>{|| t_2>t_1}{～B},

{|| t_1}{A},

{|| t_2}{B}

}

式 2

式 2 中，根據 R_2，符合前綴的 {|| t_1}{A} 應該在 t_2 時經由演序達到有 t_2 前綴的域 {|| t_2}{～B}，但是域中卻有 {|| t_2}{B}，由於前綴相同，將 {|| t_2}{～B} 與 {|| t_2}{B} 合併會（在 ～B 與 B 還沒有相消的時候）產生真值之間的矛盾「B^～B」，這種後來的事象與「先前的事象根據演序做出的推論」之間產生矛盾的情形，就稱為「動三角矛盾」。

這種合併的過程與域的表記一節中對域之間「+」符號運作方式的描述不同：{|| t_2}{~B} + {|| t_2}{B} 會導致 ～B 與 B 相消，得到既沒有 ～B 也沒有 B 在內的 {|| t_2}{}，因為單看這樣的算術操作，並沒有決定應留下 ～B 還是 B 其中一者的方式，只能使得兩者都沒有真值。而在本章介紹的學習過程當中，系統進行「矛盾消解」時，則根據本身的狀態，除了 {|| t_2}{} 這種結果以外，也可以經過特定過程後，選擇留下 {|| t_2}{～B} 或 {|| t_2}{B}，這是學習機制另外導致的可能結果，並不影響域之間使用「+」符號的操作方式。

動三角矛盾消解的方法有這三種：

- 假設非 {b}{{|| t_1}{A}}
- 假設非 {b}{{|| t_2}{B}}，即非 {b}{{|| t_2}{～B}}
- 假設非 {b}{R_2}，即非 {b}{ {|| t_1}{A} >>{|| t_2>t_1}{～B} }

在靜三角矛盾消解的討論中，我們已經知道當 {A} 是抽象域時，可能出現用來推論的前提並非「合法前提」的情況，這屬於擴張錯誤。

當前提是「合法前提」時，可以考慮真值 {b}{{|| t_1}{A}}、{b}{{|| t_2}{B}}、{b}{R_2} 的確定度何者較低，或者 {b} – {{|| t_1}{A}}、{b} – {{|| t_2}{B}}、{b} – {R_2} 何者的「穩定度」較高，比如 {b} – {{|| t_1}{A}} 的穩定度最高時，就假設非 {b}{{|| t_1}{A}}，以此類推。當三者的穩定度皆相同時，再視何者在域中的「權重」較低而定，當 {|| t_1}{A} 的權重最低時，假設非 {b}{{|| t_1}{A}}。

9.3 後撤堆疊

在上一節中，我們討論了透過移除真值來消解矛盾的方法，與存在擴張錯誤的情形。

為了減少消解矛盾時的耗能，如同行動一章中提過的計畫堆疊一樣，我們可以設定一個「堆疊」，並在特定的啟動條件滿足時，按照堆疊當中的順序嘗試調整域中的真值以進行矛盾消解，這個過程不一定可以使得到的結果最為「正確」，但可以有效地避免表示域中的矛盾消解這種耗費大量時間的過程，導致其它推論受到阻礙的情形發生。

後撤堆疊使用了一個堆疊來告訴我們矛盾三角中的真值應該如何調整，這樣的「調整」不一定只是將真值從域中移除而已，更多時候，是將會產生矛盾的真值「替換」為一個或一些相關的、但不產生矛盾的真值。後撤堆疊的形成，是透過系統在過去經驗當中遇到的矛盾情形與其後矛盾解消後的情形（不一定僅是移除真值）間的比較來達成的，具體的例子可見接下來的討論。

－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－

考慮以下的靜三角矛盾形式：

{a} {A, C=～B, R_1=A -> B}

檢視這樣的一個實例：

{a} {

A=橘子.是.橘色

R_1=1/物.是.2/顏色 -> ～(1/物.是.3/顏色)

B=～(橘子.是.綠色)

C=橘子.是.綠色=～～(橘子.是.綠色)

}

上面的例子裡，由於 R_1 的確定度很高，因此優先假設 {a}{～A} 或 {a}{～C}。

由於 A 和 C 很可能皆是由感官輸入的資訊得到，有幾個錯誤方向：

- 使 A 成立的觀察中，「橘子」實體並不是橘子（擴張錯誤）

- 使 A 成立的觀察中，「橘子」實體並不是橘色（真值錯誤）
- 使 C 成立的觀察中，「橘子」實體並不是橘子（擴張錯誤）
- 使 C 成立的觀察中，「橘子」實體並不是綠色（真值錯誤）

如果以上四個錯誤方向不成立，我們繼續考慮以下兩個方向：

- A 不成立，即「橘子是橘色」不成立（真值錯誤）
- C 不成立，即「橘子是綠色」不成立（真值錯誤）

比起嘗試依據上方列出的錯誤方向將 A 或 C 真值從域中移除，我們可以考慮使用後撤堆疊 Stack A：

Stack A [

{#啟動}{1/物.是.2/顏色, 1/物.是.3/顏色}

1/物.是.2/顏色,

>有些.1/物.是.2/顏色

]

A^2 的上標 2 跟 A_2 的下標 2 是類似的意思，A^3、A^4、… 亦同。接著得到：

{a} {

A^2=有些.橘子.是.橘色

R_1=1/物.是.2/顏色 -> ～(1/物.是.3/顏色)

B=～(橘子.是.綠色)

C=橘子.是.綠色=～～(橘子.是.綠色)

}

即

{a} {

A^2

R_1=A -> B

C=～B

}

這樣一來，矛盾就解消了。堆疊當中的 {#啟動} 是堆疊的啟動域，使系統知道應該使用眾多後撤堆疊中的哪一個；標示 A^2 是 A 的一個後撤結果，繼續後撤時，會變為 A^3、A^4、…。

－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－

再考慮域中加入 {R_2=C->D, D=～A^2} 的情形，其中「R_2=1/物.是.2/顏色 -> ～(有些.1/物.是.3/顏色)」，也就是說，「如果『一個東西是某個顏色』，『有些這個東西是另一個顏色』就不成立」，但是我們又有「C=橘子.是.綠色」，因此「A^2=有些.橘子.是.橘色」不成立，域中又產生了矛盾：

{a} {

A^2=有些.橘子.是.橘色

R_1=1/物.是.2/顏色 -> ～(1/物.是.3/顏色)

R_2=1/物.是.2/顏色 -> ～(有些.1/物.是.3/顏色)

B=～(橘子.是.綠色)

C=橘子.是.綠色=～～(橘子.是.綠色)

}

再利用另一個後撤堆疊 Stack B：

Stack B [

{#啟動}{有些.1/物.是.2/顏色, 1/物.是.3/顏色}

1/物.是.3/顏色,

>有些.1/物.是.3/顏色

]

得到

{a} {

A^2=有些.橘子.是.橘色

R_1=1/物.是.2/顏色 -> ～(1/物.是.3/顏色)

R_2=1/物.是.2/顏色 -> ～(有些.1/物.是.3/顏色)

D=～(橘子.是.橘色)

C^2=有些.橘子.是.綠色

}

此時域中無矛盾，{A^2^C^2->E}，「E=有些橘子是橘色，有些橘子是綠色」。

Stack A 與 Stack B 可以合成為後撤堆疊 Stack C：

Stack C [

{#啟動}{1/物.是.2/顏色, 1/物.是.3/顏色}

{1/物.是.2/顏色, 1/物.是.3/顏色},

>{有些.1/物.是.2/顏色.，.有些 1/物.是.3/顏色}

]

－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－

接著，看到動三角矛盾的例子：

{b} {

R_1={|| t_1}{A'} >> {|| t_2>t_1}{B'},

{|| t_1}{A},

{|| t_2}{～B}

}

A 是 A' 的實例，B 是 B' 的實例：

{b} {

R_1={|| t_1}{1/人.下雨.天.出門} >> {|| t_2}{1/人.淋濕},

{|| t_1}{小明.下雨.天.出門},

{|| t_2}{小明.沒有.淋濕},

}

此時，一個可用的後撤堆疊可能是：

Stack D [

{#啟動}{ R_1={|| t_1}{A'} >> {|| t_2>t_1}{B'}, {|| t_1}{A}, {|| t_2}{～B}}

{

R_1={|| t_1}{A'} >> {|| t_2>t_1}{B'}

{|| t_1}{A},

{|| t_2}{～B}

},

> {

R_1^2={|| t_1}{A'^X} >> {|| t_2>t_1}{～B'}

{|| t_1}{A},

{|| t_2}{～B}

}

]

得到

{b} {

R_1^2={|| t_1}{1/人.下雨.天.出門, X} >> {|| t_2}{1/人.沒有.淋濕},

{|| t_1}{小明.下雨.天.出門},

{|| t_2}{小明.沒有.淋濕},

}

此時域中無矛盾，R_1^2 代表「如果某人下雨天出門，而且 X 的話，某人不會淋

濕」，這裡 X 可能是「某人有帶傘」或其它真值。這個後撤堆疊的例子對應了「演序分岔」的概念。

－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－

後撤堆疊中，可能會有多個後撤的方案：

Stack E [

{#啟動}{1/動物.不會.飛, 2/動物個體.是.一.隻.1/動物, 2/動物個體.會.飛}

{1/動物.不會.飛, 2/動物個體.是.一.隻.1/動物, 2/動物個體.會.飛},

>{1/動物.不會.飛, {a}{2/動物個體.是.一.隻.1/動物, 2/動物個體.會.飛}},

>{1/動物.不會.飛, 2/動物個體.是.一.隻.1/動物, 2/動物個體.會.滑翔}

]

上面的後撤堆疊中，一種矛盾消解方式是增加前綴，另一種矛盾消解方式則是替換真值。當有多種後撤方案時，系統可以將最上方的方案放入假設域中，進行驗證或否證，再決定是否往下使用其它的後撤方案。

9.4 確定度、穩定域與不穩定域

在三角勾稽的矛盾消解過程中，通常可以使用後撤堆疊來快速解決域中的矛盾，然而，如果沒有相應的後撤堆疊時，就必須考慮域的「穩定度」、矛盾真值的「確定度」與「權重」來決定如何進行矛盾消解。

－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－

我們以 c(A) 來表示真值 A 的確定度，以 w_A 來表示 A 在域中的權重，確定度與權重都介於 0 與 1 之間。

當域 a 中的基底元素有 N_1、N_2、N_3、…、$N_{|a|}$，|a| 是 a 的基底數量時：

基底元素的權重和為 1：

$$w_{N_1} + w_{N_2} + w_{N_3} + \cdots + w_{N_{|a|}} = 1$$

a 的加權確定度可以表示成：

$$c(a) = w_{N_1} \cdot c(N_1) + w_{N_2} \cdot c(N_2) + w_{N_3} \cdot c(N_3)) + \cdots + w_{N_{|a|}}.c(N_{|a|})$$

當域內所有基底的確定度都為 1 時，域的加權確定度也為 1，當域內所有基底的確定度都為 0 時，域的加權確定度也為 0。

－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－

討論子域的權重時，需要先將子域化為其內基底的集合。

當 b 為 a 的子域時，b 的基底集合是 a 的基底集合的子集合。b 在 a 中的權重，是其基底集合的元素在域 a 中的權重和。

當 a 的子域 n_1、n_2、… 的基底集合共同組成 a 的基底的一個分割時（比如：{A, B, C} 和 {D, E} 共同組成 {A, B, C, D, E} 的一個分割，{A} 和 {B, C, D, E} 也共同組成 {A, B, C, D, E} 的一個分割），w_{n_1}、w_{n_2}、…的和即為 a 在 a 中的權重，也就是 1。

－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－

確定度是域中對於一個真值為真的「確定程度」，在先前的討論當中，我們通常假設域中存在一個真值時，其確定度為 1，不存在一個真值時，這個真值在域中的確定度為 0，不過，由於系統中有演序合併的過程，且矛盾真值之間也並非總是隨時得到消解，因此，真值的確定度也可能介於 1 和 0 之間。

具體而言，要決定一個真值的確定度，我們可以從域中移除該真值本身之後，使用其餘真值來構造可以導出該真值的「支持鏈」，以及構造可以導出該真值的反面的「反對鏈」，當支持鏈的確定度越高時，該真值的確定度也越高，反對鏈的確定度越高時，該真值的確定度越低。

支持鏈或否對鏈的確定度可以採用該鏈中真值確定度的最小值 N_1、N_2、N_3、…，即 $\min(c(N_1), c(N_2), c(N_3), \ldots)$，或者鏈中真值確定度的乘積 「$c(N_1) \cdot c(N_2) \cdot c(N_3) \cdot \ldots$」。當鏈上所有的真值確定度都為 1 時，鏈的確定度為 1，當鏈上的某個真值確定度接近 0 時，鏈的確定度也接近 0。

當一個真值在域中既有支持鏈也有反對鏈時，我們可以採用如 max(0, (支持鏈中確定度最大者的確定度 − 反對鏈中確定度最大者的確定度)) 這類，使真值的確定度範圍維持在 0 與 1 之間，且具有「支持鏈確定度越大，真值確定度越大」、「反對鏈確定度越大，真值確定度越小」的特質的函數來決定真值的確定度。

另外，由鏈確定度的公式如 $\min(c(N_1), c(N_2), c(N_3), \ldots)$、「$c(N_1) \cdot c(N_2) \cdot c(N_3) \cdot$

…」等，我們可以發現當構造推論鏈時，中間使用到的真值確定度過低時，會使得後續的推論結果確定度也很低。這種確定度很低的真值可能造成系統後續耗費資源在進行推論出更多確定度很低的真值上，因此，也可以採用一個推論時的「確定度閾值」，當推論鏈上的確定度降低至某個數值以下後，就不繼續往下推論。

－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－

由於真值的確定度大小由其「支持鏈」與「反對鏈」的確定度決定，當系統透過觀察獲取真值時，真值的來源的可信度會直接影響到真值的確定度。

比如，一個常說謊的智能系統說的話，或者一個可信度很低的報章雜誌中記載的內容，是較為不可信的來源，這時，從這類來源觀察到的真值，系統可能決定不以 1 為預設的確定度，而是使用與其來源的「可信程度」有關的函數來決定預設的確定度。

這樣的現象，也可以用「{小明.說.A} >> {A}」、「{小明.說.B} >> {～B}」這類的演序合併的過程，或者「{ 這.本.雜誌.裡.有.許多.謊言 -> ({這.本.雜誌}{A} >> {～A}) }」等，將真值的來源放在前綴當中，使得其中的真值向外穿透時，帶有該前綴的確定度的方式來擬合。

－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－

域中基底元素的權重是指域中有多少比例的真值的支持鏈有該真值的參與。

基底元素的權重和為 1，即 $w_{N_1} + w_{N_2} + w_{N_3} + \cdots + w_{N_{|a|}} = 1$，如果域中有三個基底元素 {$N_1$、$N_2$、$N_3$}，且域中的真值有 {$N_1$、$N_2$、$N_3$、A、B、C}，且：

- N_1 參與 N_1、A、B、C 的支持鏈（4 個）
- N_2 參與 N_2、A、B 的支持鏈（3 個）
- N_3 參與 N_3、A 的支持鏈（2 個）

那麼，我們可以選定一種方式來計算各基底元素的權重：

$$w_{n_1} = \frac{4}{4+3+2} = \frac{4}{9}、w_{n_2} = \frac{3}{4+3+2} = \frac{3}{9}、w_{n_3} = \frac{2}{4+3+2} = \frac{2}{9}$$

且使得基底元素的權重和：

$$w_{n_1} + w_{n_2} + w_{n_3} = \frac{4}{9} + \frac{3}{9} + \frac{2}{9} = 1$$

二次學習時，如果僅是將一個非基底的真值從域中移除，但不覆寫參與推論出這個真值的基底的話，後續還是會因為存在這些基底，而再次推論出同一個真值。

當要把真值 A 從域中移除時，我們可以考慮需要移除以使真值 A 不再會被推論出來的基底元素集合。這樣的集合可能有多個，且當數個基底元素參與在同一個支持鏈上以推論出 A 時，我們可以僅選擇其中權重最小，而在移除後即可使真值 A 不再被推論出來者，而不需納入全部的基底元素。

也就是說，當域 {N_1、N_2、N_3、A、B、C、…} 中的基底集合為 {N_1、N_2、N_3}，而其中 N_1 與 N_2 必須同時存在才能推論出 A，而 N_3 可以單獨推論出 A 時，要將 A 自域中移除，我們必須考慮移除「N_3」，以及「N_1 與 N_2 其中一者」。這時，移除 A 需要考慮的權重，就是 N_3 的權重，與「N_1 與 N_2 之中權重較小者的權重」的和。

由於「因為二次學習而覆寫的基底元素」如果權重很大，就代表域中確定度會受到影響的真值數量很多，也代表系統後續進行推論時，容易有受到這次調整影響的真值的參與，而必須花費額外的能量和時間重新構造出論形，因此，進行三角勾稽時，如果矛盾真值的確定度相同，我們可以進一步考慮它們對應的基底元素的權重，並優先考慮移除權重較低者。

－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－

域的穩定度與其中存在的矛盾相關，穩定度介於 1 與 0 之間。如果域中不存在矛盾，其穩定度為 1，如果域中僅存在一個矛盾三角的話，其穩定度為 0。

矛盾造成域穩定度下降的程度，與它「多重要」、「多難解消」有關。

當矛盾三角 {A, B, R_1} 的三個真值之間確定度相差越小的時候，矛盾越難解消，反之，確定度相差越大的時候，越容易解消，因此，我們加上因子：

$$|c(A) - c(B)| \text{、} |c(B) - c(R_1)| \text{、} |c(A) - c(R_1)|$$

上面三個因子當中前後的 ‖ 是取絕對值的意思。

同時，當矛盾三角 {A, B, R_1} 的三個真值在域中都很「重要」時，矛盾在域中較為重要，反之，三個真值都很「不重要」時，矛盾在域中就不重要，我們可以將真值對應的基底元素的權重視為其「重不重要」，因為當該真值必須從域中移除，使得對應的基底元素需要被覆寫時，當基底元素的權重很大，對於域中

真值的影響就很大。因此我們加上因子：

$$w_A + w_B + w_{R_1}$$

其中，w_A 代表的是 A 對應的基底元素的權重，依此類推。

由於矛盾造成域穩定度下降的程度，「同時」與其「重要性」及「難以解消的程度」有關，我們可以考慮使用對應這兩個因素的因子的乘積來表示：

$$\left(w_A + w_B + w_{R_1}\right) \cdot [1 - \max(|c(A) - c(B)|, c|(B) - c(R_1)|, |c(A) - c(R_1)|)]$$

其中，因為矛盾真值間最大的確定度差異「越大」，矛盾越容易解消，差異「越小」，矛盾越不容易解消，因此我們 [1 − 最大確定度差異] 來作為乘積當中的因子。方便起見，我們將這個乘積稱為「矛盾權重」。

為了避免域的穩定度降到 0 以下的情形，我們可以分別推算參與到每個矛盾當中的基底元素，其各自參與的最大矛盾之間，$[1 - \max(|c(A) - c(B)|, c|(B) - c(R_1)|, |c(A) - c(R_1)|)]$ 一項的值最大者，乘上這個基底元素本身的權重後，再將所有得到的值取和，最後，以「1 − 前述的和」來表示域的穩定度。

由於 $[1 - \max(|c(A) - c(B)|, c|(B) - c(R_1)|, |c(A) - c(R_1)|)]$ 的值介於 0 與 1 之間，當這項為 1 時，乘上基底元素本身的權重即為基底元素權重本身。當所有域中的基底都參與到這種難以解消的矛盾當中時，取和的過程相當於將域的所有基底權重相加，其和為 1，由「1 − 前述值為 1 的和」，我們可知域的穩定度為 0。

－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－

回顧「三角勾稽」一節，我們提過靜三角矛盾 {a} {R_1=A -> B, A, C=～B} 的消解的方法有這三種：

- 假設非 {a}{A}
- 假設非 {a}{C}，即非 {a}{～B}
- 假設非 {a}{R_1}，即非 {a}{A -> B}

至於使用三種方法中的哪一種，可以先視 {a} 中 A、C、R_1 三者的確定度而定，並考慮優先移除其中「確定度最低者」。

當三者的確定度相當時，可以再視 {a} – {A}、{a} – {C}、{a} – {R_1} 何者的「穩定度」較高而定，並優先移除「移除後穩定度最高者」。

當 {a} – {A}、{a} – {C}，與 {a} – {R_1} 的穩定度相當時，可以再視 A、C、R_1 何者在域中對應基底元素的「權重」較低而定，這是因為移除對應的基底元素權重較低者，需要覆寫的基底元素權重較低，影響到的域中真值較少，在無法透過其它方式決定移除矛盾真值中何者時，這種做法可以降低系統後續的推論負擔。

9.5 假說與為了學習的行動

假說是無法從表示域當前的基底當中推論出來的真值，系統可以透過在真值上加上前綴來在避免形成矛盾的情況下，使用假說進行推論，在尚未得到驗證的假說上加上前綴的過程，可以透過將假說真值放在「假設域」中來達成。

假設域是系統表示域的一個子域，表示域中，在假設域以外的真值可以穿透進入假設域中，但假設域中的假說真值則無法向外穿透。當系統經過推論或者「為了學習的行動」，也就是可以驗證或否證假說的行動而增加假設域外的真值，使得假設域外的真值也能推論出該假說時，假說就進入到表示域中假設域外的部分。

系統透過觀察形成的演序規則時常發生「分岔」，比如我們常舉的例子「1/人.下雨天.出門 >> 1/人.淋濕」和「1/人.下雨天.出門 >> 1/人.沒有.淋濕」，這裡在矛盾消解的後撤堆疊當中，可能使用「{1/人.下雨天.出門 ^ ～X} >>{ 1/人.淋濕}」與「{1/人.下雨天.出門 ^ X} >>{ 1/人.沒有.淋濕}」來解決域中的矛盾，也就是透過「分岔因子」來避免使用不同的演序規則推論出矛盾結果。

X 可能是「1/人.有.帶.傘」，但是表示域中可能既沒有 X，也沒有 ～X，此時，將 X 放入假設域中，系統並進行推論或安排可以驗證或否證 X 的行動，使得表示域中出現 X 或 ～X，就可以透過後撤產生的規則來完成無矛盾的推論。針對假設域中的值進行推論或行動安排的行為，可以透過使推論與行動的掃描清單中也包含假設域中已啟動的真值來達成。

－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－

在範式當中，可能存在規則大致相同，但是有「關鍵變因」使得範式中的真值需要進行廣泛調整的情形，比如看到這個例子：

{a_1}{

{#啟動}{把手.向.下.調整.水量.會.變.小, 把手.向.上.調整.水量.會.變.大, 把手.向.左.調整.水溫.會.變.熱, 把手.向.右.調整.水溫.會.變.冷}

{溫度.與.水量.調整}{|| X_1}

{

水量.太.大.時.要.把.把手.向.下.調整

水量.太.小.時.要.把.把手.向.上.調整

水溫.太.熱.時.要.把.把手.向.右.調整

水溫.太.冷.時.要.把.把手.向.左.調整

把手.向.下.調整.時.，.水溫.會.變.熱

水量.太.大.而.水溫.剛好.時.，.要.把.把手.向.右.下.方.調整

}

}

另一個類似的域是：

{a_2}{

{#啟動}{把手.向.下.調整.水量.會.變.小, 把手.向.上.調整.水量.會.變.大, 把手.向.右.調整.水溫.會.變.熱, 把手.向.左.調整.水溫.會.變.冷}

{溫度.與.水量.調整}{|| X_2}

{

水量.太.大.時.要.把.把手.向.下.調整

水量.太.小.時.要.把.把手.向.上.調整

水溫.太.熱.時.要.把.把手.向.左.調整

水溫.太.冷.時.要.把.把手.向.右.調整

把手.向.下.調整.時.，.水溫.會.變.熱

水量.太.大.而.水溫.剛好.時.，.要.把.把手.向.左.下.方.調整

}

}

上面加上前綴 X_2 的域中，畫底線的部分僅是為了使讀者容易看出與前綴 X_1 的域中哪裡不同。

像這樣，不同的範式形式類似，僅因少數的關鍵變因而使得真值需要調整，且需要調整的真值在範式間互相矛盾的情形，我們將這些範式稱為「姐妹範式」。我們可以參考含有姐妹範式的域的啟動條件，將同樣的啟動條件設定在可以驗證或否證關鍵變因的推論或行動定形上，這樣一來，系統就可以在將關鍵變因的一個「候選真值」設定為假說後，快速進行假說的驗證或否證，以決定採用姐妹範式中的哪一個來組成遊戲策略，這個例子當中，遊戲可能是「洗澡.時.的.溫度.與.水量.調整」，目標域可能是「使水量與水溫剛好」。

－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－

在具有「解釋真值發生的原因」傾向的系統當中，系統透過生成逆推論鏈來形成演序，然而，逆推論鏈上可能發生分岔，即域中有 {C, R_1=A>>C, R_2=B>>C} 的情形。

這時，發生目前事象的原因在驗證假說之前，對於系統而言具有機率性，如果 R_1 的確定度更高，系統可能會認為 A 為 C 的原因的「機率更大」（即使我們知道數學上這樣並不嚴謹），如果 R_2 的確定度更高，則系統可能認為 B 為 C 的原因的機率更大，如果兩個規則的確定度相同，可能可以進一步參考權重。

看到以下的例子：

{

小明.遲到,

R_1={1/人.在.路上.遇到.塞車} >> {1/人.遲到},

R_2={1/人.太.晚.出門} >> {1/人.遲到}

}

假設 R_1 和 R_2 的確定度與權重都相同，系統就無從判斷「遇到塞車」和「太晚出門」何者為小明遲到的原因。

這時，可以將一個「候選真值」（這裡是「小明.在.路上.遇到.塞車」或「小明.太.晚.出門」）設定為假說，並且安排能夠驗證或否證該假說的行動，比如「詢

問小明路上是否有塞車」，來增加表示域中的真值並試圖以此解釋真值「小明.遲到」發生的原因。

－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－

在系統的推論定形當中，可以存在一類定形，經過這類定形對應的定式作用，規則的使用範圍會「向上擴張」或「向外擴張」。

「擴張假說」就是將規則的使用範圍擴張所形成的假說，其中，「抽象擴張」是將規則中的實體往上層進行抽象化，「轉換擴張」則是將規則中的實體替換為非其上層分類所涵蓋的實體。比如，將規則中的「/昆蟲」以「/動物」替換，屬於「抽象擴張」，將規則中的「/自然人」以「/法人」替換，屬於「轉換擴張」。

我們可以將「設定假說」，也就是在表示域的子域「假設域」中添加真值，視為是在表示域中增加一個開始時確定度為零的真值，接著，可以透過推論定形或觀察帶來的新的真值來增加假說的確定度。

因此，經由抽象擴張或轉換擴張而設定的假說，可以經過構造表示域中的支持鏈來增加其確定度，當確定度超過一定的閾值時，假說就得到「驗證」，當表示域中的反對鏈使「假說的反向」的確定度超過一定的閾值時，假說就得到「否證」。

9.6 不學習

當學習所需要耗用的能量巨大時，系統可能使用以下方式來避免能量耗用：

- 從輸入資訊當中篩除
- 進入表示域後遺忘
- 在真值上添加不必要的前綴

這些導致「不學習」的方式雖然能夠降低系統耗能，但在初次學習與二次學習當中，新真值客觀來說「為真」時，對於系統對其環境的預測力就會產生負面影響，在二次學習當中，也會使得錯誤的基底真值及其推論結果無法被排除。

然而，「不學習」並非在所有情況下都具有負面影響。在錯誤資訊密集或者當下環境的狀況特殊，真值與系統所處的多數環境牴觸時，「不學習」也可能是有效率，甚至能夠減少後續的推論錯誤。

不同的智能系統具有不同程度的「低耗能傾向」，當系統的推論能量充足時，系統的低耗能傾向就較弱，也更有可能形成並執行許多有助於檢視及排除錯誤真值的定形。相對的，當系統因屏蔽閾值很高，比如處於危險、壓力、資源匱乏當中時，低耗能傾向較高，檢視及排除錯誤真值的定形較少執行，偶然發生學習時，也更有可能使用前述的「篩除」、「遺忘」、「不必要前綴」等方法來避免學習。

－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－

二次學習耗用的能量與其覆寫真值的確定度有關，新真值覆寫的真值確定度越大，新真值的「學習摩擦」也隨之越大，這是因為原有真值的確定度之所以高，是因為它有很多確定度高的支持鏈，替換為新真值時，容易與這些鏈上的節點之間產生矛盾，也使得表示域的穩定度下降。

同時，當原有真值參與的推論鏈與行動方案眾多時，移除原有真值也可能產生巨大的「推論債務」：移除所造成的論形上斷鏈或行動方案不再適用，會使後續又進行相關的推論或安排行動方案時，需要經過重新構造推論鏈與行動鏈的過程而花費額外的能量，推論需時也更長。

因此，如果系統被設定為學習摩擦超過較低的閾值就不學習，雖然代表二次學習會因此較難發生，但在某些時候，這反而是有效率的。

另外，也要考慮到系統的所在環境或學習內容的特性，有時系統所在環境中容易觀察到錯誤或隨機的真值，觀察結果可能反反覆覆導致推論債務反覆發生，影響系統的推論及行動速度。另外，有時系統學習到的新真值其正確的前綴對於系統的推論或行動而言很少適用，此時新真值使系統增加的預測力與減少錯誤的效益很可能無法使額外耗費的記憶容量與推論能量顯得值得。

當然，無論是初次學習或二次學習，系統都可以透過在新真值前方加上額外前綴而避免真值覆寫的發生，以此來完全避免「不學習」的情形，不過，這會耗費額外的記憶容量且可能增加後續推論時的耗能，因此，對於記憶容量很大、推論能量充分的系統而言，總是加上前綴儲存新資訊的作法可能更好，而相對的，對於沒有這些特質的系統而言，在學習摩擦較大時，可能選擇「不學習」是更適當的選項。

綜合來說，系統的學習摩擦閾值不僅因個體而異，即使是同一個系統，在不同環境當中使用不同的閾值，也更可能達到更好的效果。

9.7 元學習

元學習是指並非以語句「是」或「否」為核心，而是以行動或狀態「好」或「不好」為核心進行的系統最佳化過程。

什麼樣的情況下「要做什麼行動」或者「不要做什麼行動」，也可以表達為「什麼樣的情況下做什麼行動是『好』的」，以及「什麼樣的情況下做什麼行動是『不好』的」。什麼樣的情況下要做什麼、不需要做什麼、要想什麼、不需要想什麼，這些與「行動定形」和「推論定形」入口域的啟動條件有關的最佳化過程，就是元學習的一種。

另外，效益函數中的因子，「賺很多錢是好的」、「身體健康是好的」、「幫助很多人是好的」、「肚子餓不好」、「處在危險的環境不好」等等，也是以「好」與「不好」為中心的最佳化。

－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－

我們在「不完全系統的議題」一章中對於「發展階段」的討論，已經涵蓋了系統進行的「學習」與「元學習」。

在系統的形成階段，系統會發生「真值添加」以及「運行與簡化」，真值添加就是本章提到的「初次學習」。

「運行與簡化」中：

- 「矛盾消解」屬於二次學習
- 「否證真值」屬於二次學習
- 「定形的形成」與是或否無關，而與好或不好有關，屬於元學習
- 「行動計畫的選擇與執行」既與初次與二次學習形成的真值有關，也與元學習形成的定形有關

系統的最佳化部分：

- 行動方案的增加、增加記憶真值數量的方法（學習特定的記憶技巧）等，是透過真值的增加來達成的，屬於初次學習
- 調整定形組合、調整掃描順序、調整屏蔽閾值等等，以及情緒的調節，屬於元學習
- 減少錯誤的部分，與「應該」怎麼做、「不應該」怎麼做的觀察技巧、推論技巧有關，屬於元學習

元學習就是透過觀察系統本身執行了什麼、想了什麼，並評估這些執行的行動是否帶來（透過效益函數評估）好的結果，進行的推論過程是否得到有用（比如符合原本設定要得到的形式）的結論等，來為這些推論與行動的決定過程「加減分」或作調整。另外，如果已經忠實按照原先效益函數的設定，成功執行了許多行動，到達許多設定好的目標域，卻發現後續的演變在納入更多事象來檢視時，根據效益函數計算出來的結果不如理想，此時，也可能透過元學習來調整效益函數中的某些因子的權重，以期未來能夠設定出更適當的目標域。

綜合來說，「學習」影響的是系統認知中「什麼是對的」、「會發生什麼」，「元學習」則與「做什麼是好的」、「什麼狀態是好的」有關，調整的是推論與行動發生過程的機制，包含效益函數的設定。透過學習與元學習，系統可以認知到正確的真值，並調整其行為以更接近最佳化系統，或者透過改變效益函數的途徑來影響自身認知到的效益。

第十章：錯誤

10.1 錯誤的兩種類型

智能系統在「資訊輸入」、「進行學習與推論」、「安排行動」的整個過程當中，都有可能出現錯誤。根據錯誤本質的不同，我們將其區分為「任意宇域的錯誤（絕對錯誤）」與「非任意宇域的錯誤（相對錯誤）」，以表示該錯誤與系統所處的環境有關或者無關，只要判斷不屬於「任意宇域的錯誤」，都歸為「非任意宇域的錯誤」。

錯誤不包含「不適當/不好」、或者「非最佳化」的情形，也就是說，一個智能系統安排並執行違反規範（法律、倫理、特定信念）的行為，不在我們定義的錯誤範疇內；同時，「沒有效率的推論過程」、「安排了很難達成目的的行動」等，屬於系統的最佳化問題，這些我們也不將其視為「錯誤」。

－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－

「任意宇域的錯誤」是指該錯誤在任何一個環境（世界）當中，只要該環境與我們現處的世界具有一樣的「推論邏輯」，那麼這個錯誤就必定仍然是一個錯誤，而不會是正確的，舉一個最簡單的例子：

$$\{R_1=A->B, A\}$$

如果在上述的域中，使用 R_1 規則與 A 的前提，卻推論出 ～B，就屬於任意宇域的錯誤（宇域是指當前系統所處的環境的超域再往外推至超域，直到最大可考慮的範圍）。為了討論時的方便與減少混淆情形，我們也可以把這類錯誤簡單稱為「絕對錯誤」。

絕對錯誤是「過程」的錯誤，它代表推論在過程中就發生錯誤，而不論其推論結果與系統所處的環境中的事象是否相符，由於推論的過程本身有錯誤，系統難以透過二次學習來將其修正，且這類錯誤是可避免的，因此若建構的人工智能系統中發生這類錯誤，需要積極進行分析與調整。

「非任意宇域的錯誤」是指該錯誤雖然與系統現處環境中的事象衝突，但是可能存在某個環境（世界），可以使得該錯誤不是一個錯誤，也就是說，可能在該另一個環境下，系統內的這個真值與所處宇域中的真值間就不會發生衝突，比如：「大多數的建築物都是圓球形」這個敘述可能對於系統所處的環境而言不是

真的，在這個環境中，是一個「真值錯誤」，然而，也可能有另一個環境（世界）使得這個敘述是真的，在該環境中它就不是一個錯誤的真值。

像這樣，在某些宇域中是錯誤，在其它一些可能存在的宇域中卻不是錯誤者，我們就將其稱為「非任意宇域的錯誤」，方便起見，也稱為「相對錯誤」。

相對錯誤是「結果」的錯誤，代表推論過程本身沒有發生錯誤（如果過程有錯誤，屬於「絕對錯誤」），但是產生了與系統所處的環境中的事象並不相符的真值。由於推論的過程本身沒有錯誤，只是真值有錯誤，系統比較容易透過二次學習將其修正。由於包含域的抽象化、假設真值等過程都可能導致相對錯誤，這類錯誤幾乎無法避免。人工智能系統中發生這類錯誤屬於常態，而且包含所有人類個體在內的智能系統，基本上時時都存在著比例或多或少的相對錯誤。

10.2 非任意宇域的錯誤 – 相對錯誤

相對錯誤是指推論「結果」中發生錯誤，由於推論的結果是一個真值的集合，因此相對錯誤必定會導致「錯誤」的真值；所謂「錯誤」的真值，指的是這個真值與系統所處環境的事象不相符合。當我們說發生了「真值錯誤」，就是發生了這種正確推論過程產生與環境不符合的「錯誤」真值的情形。

本身過程中沒有出錯，但是結論中有真值錯誤的推論過程，不是一種「過程」中的錯誤。許多時候，結論中有真值錯誤的原因是前提或推論規則本身也是錯誤的真值，經過了合法的推論過程後，得到同樣屬於錯誤的真值。這種情形相當常見，甚至屬於多數智能系統中不可避免的常態，也是系統必須存在二次學習機制的理由。

除了上面提到的這種前提或推論規則本身也是錯誤真值的情形以外，還有另一種雖然合法，但同樣導致結論中有錯誤真值的「過程」，也就是「擴張錯誤」。擴張錯誤發生在系統將觀察到的事象實例透過抽象過程形成抽象真值的過程當中，由於經過抽象後（原本沒有斜線標記的真值中加入斜線標記，或者斜線標記的抽象層級向上或向外擴張），就會適用到更多的實例上，如果採用了太高的抽象程度，或者複用到並沒有這個相同特質的一組實體上，就會導致套用到一些新適用的實體上時產生錯誤的真值。

以下的例子中，我們先看到「真值錯誤」，再看到「擴張錯誤」是如何導致真值錯誤的。

另外，「假設錯誤」指的是系統透過定形假設了錯誤的真值，使得表示域中產生真值錯誤。由於假設出的真值幾乎總是由規則的擴張適用而來，因此可以視為擴張錯誤的一個子類。

－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－

已經知道：

- 小茜、小明、小華、小美是同一所高中（RT 高中）裡的四位同學
- 小茜和小明就讀一年級 Alpha 班
- 小華就讀一年級 Beta 班
- 小美就讀二年級 Beta 班

小茜知道以上的所有事實。

另外：

- 「機智小貓探險」是一部有趣的生態教育影片
- 只有一年級的所有學生（含 Alpha 班與 Beta 班）看過「機智小貓探險」
- 小茜和小明同班，一起在班上看了「機智小貓探險」，因此知道小明和自己都看過
- 小茜沒有「看到」小華和小美看過，但是由於自己和小明在學校看過「機智小貓探險」，因此認為他們也看過

給定上述的前提，小茜作為一個智能系統的表示域中有：

{小茜}{

A.=.小茜.看.過.「.機智.小貓.探險.」

B.=.小明.看.過.「.機智.小貓.探險.」

C.=.小華.看.過.「.機智.小貓.探險.」

D.=.小美.看.過.「.機智.小貓.探險.」

}

由於小美事實上沒有看過「機智小貓探險」（根據我們的假設，這是絕對的事實），所以對於作為智能系統而處於其環境當中的小茜而言，真值 D，即『小

美.看.過.「.機智.小貓.探險.」』是一個錯誤的真值。

－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－

再看到「擴張錯誤」導致真值錯誤的過程：

{小茜}{

A.=.小茜.看.過.「.機智.小貓.探險.」

B.=.小明.看.過.「.機智.小貓.探險.」

C.=.小華.看.過.「.機智.小貓.探險.」

G. =.小茜.是.RT.高中.的.學生

H. =.小明.是.RT.高中.的.學生

I. =.小華.是.RT.高中.的.學生

K.=.小美.是.RT.高中.的.學生

} >>{小茜}{

A, B, C, G, H, I, K,

L'.=./RT_高中_的_學生.看.過.「.機智.小貓.探險.」

M.=.小美.看.過.「.機智.小貓.探險.」

}

上面的例子當中，小茜由 A、B、C 抽象得到的

A'.=./RT_高中_的_學生.看.過.「.機智.小貓.探險.」(=B'=C')

合併推論出 L'。

接著由 L' 與 K，推論出錯誤真值 M，即『小美.看.過.「.機智.小貓.探險.」』。

我們已經知道 M 是一個錯誤的真值。這種透過向上抽象過程形成的抽象規則，套用到其它實例上而產生真值錯誤的過程，就是擴張錯誤。

－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－

再看到一個「假設錯誤」的例子，由於假設出的真值幾乎總是經由擴張過程而來，我們可以將其視為擴張錯誤的一個子類：

{小茜}{

B.=.小明.看.過.「.機智.小貓.探險.」

{<小明>‖ 在.家.裡}{}

} >> {小茜}{

B.=.小明.看.過.「.機智.小貓.探險.」

{<小明>‖ 在.家.裡}{E.=.看.過.「.機智.小貓.探險.」}

}

而我們事實上知道小明只在學校看過「機智小貓探險」，而沒有在家裡看過。但是小茜可能因為自己在學校看過了這個影片後，回家之後也看了好幾次，因此認為小明也在家裡看過。這個本身合法的假設過程產生了真值錯誤，我們將這種情形稱為「假設錯誤」，是擴張錯誤下的一種子類。

－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－

同時，我們看到小茜的表示域中，命名為 <小明> 且帶前綴 {‖ 在.家裡} 的域中，有真值 E。如果將 E 視為是真值 B 透過「穿透規則」向內穿透而來，那麼 {小茜} 域中的真值：

{<小明>‖ 在.家.裡}{ E.=.看.過.「.機智.小貓.探險.」 }

或者以沒有前綴的形式表示：

F.=.</小明>.在.家.裡.看.過.「.機智.小貓.探險.」

一樣是一個錯誤的真值。

如果穿透規則本身沒有錯，是因為 F 這個錯誤真值先透過假設過程出現在域中，再正確穿透到 {<小明>‖ 在.家.裡} 域裡，就是發生了假設錯誤。然而，如果 F 本來不在域中，是因為存在錯誤的穿透規則，使得 B 直接經過穿透規則進入到子域裡，那麼我們則稱產生子域中 E 真值的穿透過程當中發生了「穿透錯誤」。

穿透錯誤屬於本節最前面段落提到的「結論中有真值錯誤的原因是前提或推論規則本身也是錯誤的真值」的情形。這個例子中，是因為屬於推論規則的穿透規則本身是錯誤（1/人.看.過.2/影片->1/人.在.家.裡.看.過.2/影片）的。

10.3 任意宇域的錯誤 – 絕對錯誤

絕對錯誤是指推論「過程」中發生的錯誤，它與推論結果本身是否含有真值錯誤無關。當過程發生錯誤，導致出現真值錯誤時，固然屬於絕對錯誤，過程發生錯誤，但推論出的真值無錯誤時，仍然屬於絕對錯誤。

由於絕對錯誤的特性使得系統在這樣的錯誤下「越學習、反思，錯誤越多」的傾向明顯，與相對錯誤「學習、反思可以使得錯誤得到修正」的傾向更強的情形不同，因此當所欲建構的系統中出現絕對錯誤時，需要更加積極地檢視錯誤發生原因並進行調整。

絕對錯誤的類型主要有兩種：

- 觀察錯誤
- 推論錯誤

觀察錯誤指的是錯誤發生在由感官輸入的資訊形成真值之前，是感測器接收資訊或輸入介面映射過程中發生的錯誤，而與推論無關。

舉例而言：小茜看到小明牽著一隻寵物跑過去一閃而過，由於時間短暫，小茜以為小明牽著的是一隻狗，但其實是一隻貓。

小茜的表示域中可能有「小明.牽.著.一.隻.狗.跑.過去」，這個錯誤（不符合環境中實象）的真值與小茜的推論無關，而是在進行推論前即發生了錯誤，我們將這種情形稱為「觀察錯誤」。

必須注意的是，我們所設定的情境中，小茜是因為觀察時間短暫而誤認，如果觀察時間充分，小茜定睛細看，仍然認為小明牽著的貓是一隻狗，那麼很可能是小茜將那隻貓的特徵認定為是狗的特徵，屬於擴張而來的真值錯誤（從具有 A 特徵的是狗、具有 B 特徵的是狗等等實例抽象而來的規則）。

另外，如果小茜是因為認為小明只有養狗沒有養貓，但小明其實是只有養貓，且小茜又認為養寵物的人通常只會帶著自己的寵物在路上跑，因而推論出「小明.牽.著.一.隻.狗.跑.過去」這樣錯誤的真值，那麼這就屬於由錯誤前提推論而來的真值錯誤。

－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－

絕對錯誤中的推論錯誤則是指給定前提真值與規則，系統卻沒有正確依照規則得到推論結果的情形。

透過前面章節中關於推論規則的介紹，我們可以列舉出這四種推論錯誤：

- $\{a_1\}\{ R_1 = A->B, A \}\ [R_1]->\ \{a_2\}\{ R_1 = A->B, A, C \}$，且 B 不在 a_2 中。
- $\{a_1\}\{ R_1 = A->B \}\ [R_1]->\ \{a_2\}\{ R_1 = A->B, B \}$，且 A 不在 a_1 中。
- $\{ R_1 = \{A\}>>\{B\}, \{a_1\}\{A\}\ [R_1]>>\ \{a_2\}\{A, C\} \}$，且 B 不在 a_2 中。
- $\{ R_1 = \{A\}>>\{B\}, \{a_1\}\ [R_1]>>\ \{a_2\}\{B\} \}$，且 A 不在 a_1 中。

第一種是給定前提與規則，卻推論出錯誤結論，比如根據前提「今天是星期六」以及推論規則「今天是星期六的話，今天是週末」，得到錯誤結論「今天不是週末」，或者得到任何並非與「今天是週末」同構的語句，比如「貓比狗跑得快」，或者沒有推論出任何結論。

第二種是給定規則，卻在前提沒有成立的情況下推論出結論，比如根據推論規則「今天是星期六的話，今天是週末」，在沒有「今天是星期六」的前提下，得到（正確或錯誤的）結論「今天是週末」。

第三種是給定前提與演序，卻使符合前提的域中轉換為錯誤的域，比如根據前提「小明遲到」以及演序「小明遲到的話，會被罰寫課文」，得到具有錯誤真值「小明遲到且小明不會被罰寫課文」的域，或者推論出任何並非與含有「小明遲到且小明會被罰寫課文」的域同構的域，比如得到「小明遲到且小明喜歡貓」，或者只得到含有「小明遲到」這個前提的域。

第四種是給定演序，卻使沒有符合前提的域中轉換為結果域，比如根據演序「小明遲到的話，會被罰寫課文」，使沒有「小明遲到」的域轉換為具有真值「小明被罰寫課文」的域。

由於建構人工智能系統時，通常會依據正確的推論方法先行寫定與推論有關的邏輯，因此只要初始狀態正確，建構出的人工智能系統應該少有這類錯誤。

10.4 非屬錯誤的非最佳化狀態

我們將「錯誤」定義為推論的過程不合法（絕對錯誤），或者推論的結果與實象不符（相對錯誤）。

有些系統內發生的沒有效率的變化過程，既不屬於絕對錯誤，也不屬於相對錯誤，而是由於系統處於非最佳化的狀態而發生，我們不將其視為「錯誤」，而是

另外作為「非最佳化狀態」來討論。

本節中，我們討論兩種非錯誤的真值從系統中被移除的非最佳化過程：

- 矛盾消解過程的過度移除
- 擴張錯誤後的過度限縮

二次學習與遺忘是系統當中真值被移除的兩種原因，上面列出的兩種情形都是由於「二次學習」而發生，至於遺忘的機制是否有效率，則不在此節的討論範圍內。

－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－

二次學習的矛盾消解過程，可能發生過度移除的情形。回憶「學習」章節中所介紹的矛盾消解過程：

靜三角矛盾：{a} {R_1=A -> B, A, C=～B}

靜三角矛盾的消解方法有這三種：

- 假設非 {a}{A}
- 假設非 {a}{C}，即非 {a}{～B}
- 假設非 {a}{R_1}，即非 {a}{A -> B}

如果自矛盾發生前的情形開始檢視：

$$\{R_1 = A\text{->}B, A\}\ [+\sim B] >> \{R_1,\ \sim A,\ \sim B\}$$

在域 {R_1 = A->B, A} 中，由於 [+～B]，使得域成為 {R_1 = A->B, A,～B}，產生靜三角矛盾。當我們不考慮「非 {a}{～B}」的選項時，消解的方法應該為 {R_1, A,～B} – {A} 或 {R_1, A,～B} – {R_1}。

此時，如果使用 {R_1, A,～B} – {R_1, A} 來作為消解的結果，也就是同時將規則與前提都從域中移除，就發生了「過度移除」。

比如，「A=小明比小華高」、「R_1=1/人比 2/人高->2/人比 1/人矮」、「B=小華比小明矮」、「C=～B=～（小華比小明矮）」，當我們相信小華比小明矮的反面具有真值時，應該考慮 A 沒有真值或 R_1 沒有真值的情形，而非同時將 A 與 R_1 的真值都從域中移除，因為這種沒有效率的矛盾消解方式，可能使得非錯誤的真值持續在二次學習發生時持續從系統中被移除。

動三角矛盾：

{b} {

R_2={|| t_1}{A} >>{|| t_2>t_1}{～B},

{|| t_1}{A},

{|| t_2}{B}

}

動三角矛盾消解的方法有這三種：

- 假設非 {b}{{|| t_1}{A}}
- 假設非 {b}{{|| t_2}{B}}
- 假設非 {b}{R_2}，即非 {b}{ {|| t_1}{A} >>{|| t_2>t_1}{～B} }

如果自矛盾發生前的情形開始檢視：

{b_1}{

R_2={|| t_1}{A} >> {|| t_2>t_1}{～B},

{|| t_1}{A}

} [+{|| t_2}{B}]>> {b_2}{

R_2,

{|| t_1}{A} >> {|| t_2>t_1}{B}

}

在域 {b_1}{R_2, {|| t1}{A}} 中，由於 [+{|| t_2}{B}]，使得域成為：

{b_2}{ R_2, {|| t1}{A}>>{|| t_2>t_1}{B}}

而根據 R_2，則應該是：

{ R_2, {|| t1}{A}>>{|| t_2>t_1}{～B}}

產生了動三角矛盾。當我們不考慮「非 {b_2}{{|| t_2}{～B}}」的選項時，消解的方法應該為 {b_2} – {{|| t_1}{A}} 或 {b_2} – {R_2}。

此時，如果使用 {b_2} – {{|| t_1}{A}, R_2} 來作為消解的結果，也就是同時將演序規則與前提域都從域中移除，同樣發生了「過度移除」。

比如：

{

{|| t_1}{A=小明吃兩碗飯}

R_2={|| t_1}{1/人.吃.兩.碗.飯} >> {|| t_2>t_1}{～（1/人.想.再.吃.一.碗.飯）}

{|| t_1}{A=小明吃兩碗飯} >> {|| t_2}{B=小明想再吃一碗飯}

}

當我們相信「小明想再吃一碗飯」具有真值時，應該考慮某個特定的先前時間點「小明吃兩碗飯」沒有真值或規則 R_2 「如果一個人吃了兩碗飯，他就不會想再吃一碗飯」沒有真值的情形，而非採取同時將兩者都從域中移除這種沒有效率的矛盾消解方式。

－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－

「擴張錯誤後的過度限縮」可以使用「相對錯誤」一節中關於擴張錯誤的例子來接續說明，如果沒有印象，讀者可以重新參考該例子。

- 小茜、小明、小華、小美是同一所高中（RT 高中）裡的四位同學
- 小茜和小明就讀一年級 Alpha 班
- 小華就讀一年級 Beta 班
- 小美就讀二年級 Beta 班
- 只有一年級的所有學生（含 Alpha 班與 Beta 班）看過「機智小貓探險」

假設後續小茜發現～M，也就是小美沒有看過「機智小貓探險」後，進而將 L’ 修正為抽象階層較低的 N’：

{小茜}{

A.=.小茜.看.過.「.機智.小貓.探險.」

B.=.小明.看.過.「.機智.小貓.探險.」

C.=.小華.看.過.「.機智.小貓.探險.」

G, H, I, K,

～（M.=.小美.看.過.「.機智.小貓.探險.」）

N’.=./RT_高中_一_年級_Alpha_班_的_學生.看.過.「.機智.小貓.探險.」

} >>{小茜}{

A, B, ~~C~~, G, H, I, K,

～（M.=.小美.看.過.「.機智.小貓.探險.」），

N’

}

由於原先小茜採用了更高抽象層級的：

L’.=./RT_高中_的_學生.看.過.「.機智.小貓.探險.」

現在則修正為較低抽象層級的：

N’.=./RT_高中_一_年級_Alpha_班_的_學生.看.過.「.機智.小貓.探險.」

這樣一來，『小華.看.過.「.機智.小貓.探險.」』的真值失去支持鏈，可能從系統中被移除，然而，根據假設，實象更接近真值 Q：

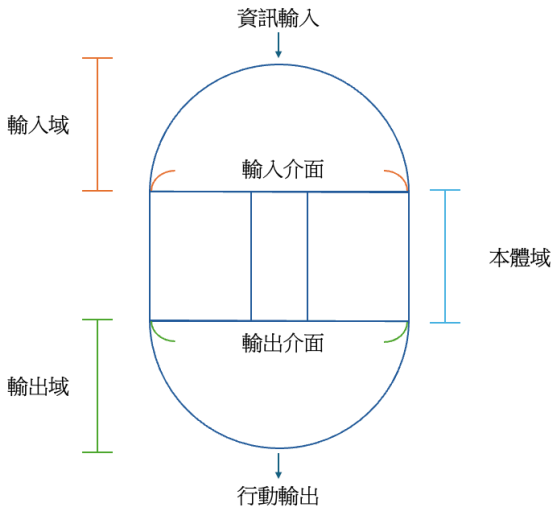
Q’.=./RT_高中_一_年級_的_學生.看.過.「.機智.小貓.探險.」

而且同樣根據假設，小華實際上是有看過「機智小貓探險」的。

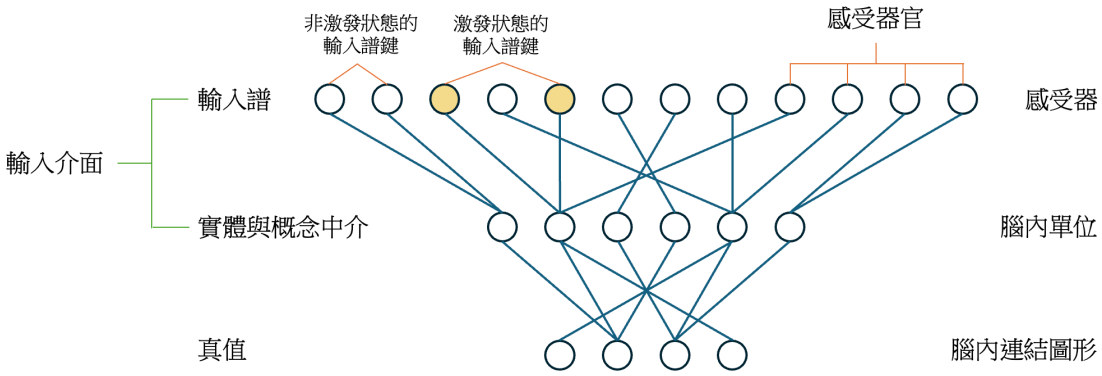
這代表小茜一開始透過真值的抽象過程（小茜自己和小明都看過，而向上抽象至「/RT_高中_的_學生」層級）將其適用範圍擴張到小華與小美身上，後續卻因為發現小美並沒有看過，所以限縮適用範圍到「/RT_高中_一_年級_Alpha_班」的層級，因而使在「/RT_高中_一_年級_的_學生」適用範圍內的小華不再處於實例範圍內。

這種擴張適用範圍後，因為二次學習而導致的過度限縮，由於只是移除了系統中的非錯誤真值，而沒有新增錯誤的真值，因而不屬於相對錯誤，同時，由於採用的限縮沒有「不合法」，而只是「非最佳化」，因而同樣屬於「非最佳化狀態」的情形。

11.1 感官系統



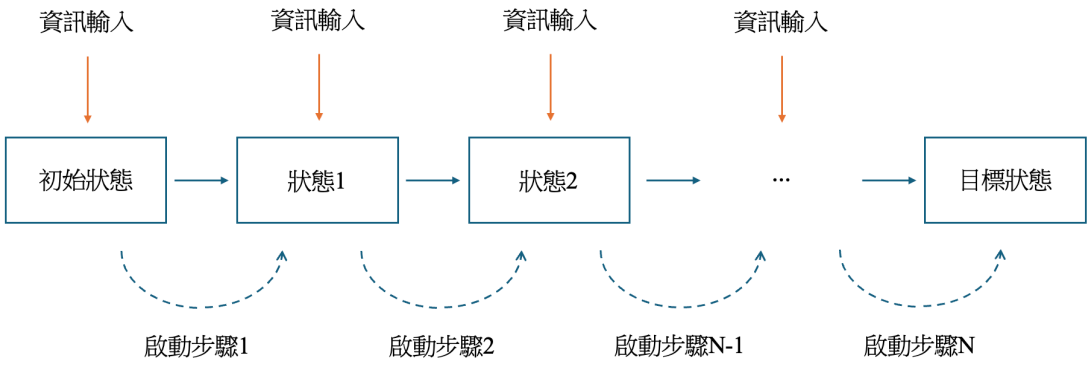
感官是資訊輸入系統的管道，感官可能是電子、機械、生物結構的感測器，接收系統所在環境的電磁波、聲波、力、熱、化學分子等，並將其透過映射過程轉化為系統「輸入譜」上的狀態。



如上圖所示，感官（感受器官）將輸入訊號轉換為輸入譜上的狀態（某些譜鍵正受到激發，其餘譜鍵並未受到激發），輸入譜上的單位（即譜鍵）對於「實體或概念中介的單位」的映射，就是系統的「輸入介面」。複數個實體與概念中介的單位進一步形成真值，而真值是系統內進行推論的單位。

上圖中，也列出了「輸入譜」、「實體與概念中介」及「真值」在人類個體作為智能系統上的對應，以方便理解。有關人類與其他生物的感官結構細節，屬於神經科學的探討範疇。

11.2 初始狀態與系統啟動



使用高階方法建構智能系統時，需要設計出「初始狀態」與「啟動過程」，其中，初始狀態涵蓋了感官、輸入譜與輸出譜、記憶機制、學習機制、注意力機制等，主要透過硬體實現的系統功能。

以人類而言，初始狀態中涵蓋的感官就是我們身體的五官、皮膚等處與其中含有的感受器，當設計人工智能系統時，這些感受器可以使用機械或電子結構來替代，只要可將環境中的電磁波、聲波、力等訊息轉換為輸入譜即可。

輸入譜與輸出譜以人類而言，即是由各個身體部位傳遞輸入訊號到大腦、以及由大腦傳遞輸出訊號到各個身體部位的神經連結。外界資訊以輸入譜形式傳遞到大腦的相應部分後，會經過「輸入映射」的過程而成為以大腦中神經連結的特定形式而表示的「真值」，而大腦的其它部位安排行動後，也會經過「輸出映射」的過程而成為向身體各部位傳送出的神經訊號，進而觸動肌肉等器官而移動身體。

記憶機制以人類而言，是大腦中神經細胞增加、產生連結而形成特定形式以儲存真值，而學習機制中的二次學習則與神經細胞的減少，或者連結的改變或減弱、消失有關。注意力機制與身體的能量管控，以及同時間大腦中能夠透過電流或化學形式而呈現激發狀態的神經連結數量有關。

上述的這些機制，在人類個體這樣的智能系統上都是透過身體中的硬體機制而實現的，因此其實現方式並不是高階理論注重探討的範圍（其功能的呈現結果則在探討範圍內）。在以高階理論建構人工智能系統時，應先述清初始狀態中各部分需實現的功能呈現結果，並使用其它學門的技術將其建立出來，無論使用的是機械方法、電子方法，還是生物學方法皆可。

有了初始狀態後，需要注重討論的將是人工智能系統的「啟動過程」。啟動過程

是在系統初始狀態與其後的系統狀態上的一系列資訊輸入，這些有序的資訊輸入將與初始狀態中的機制交互，使得系統的狀態在一連串的改變後到達目標狀態。

我們可以將啟動的過程看作是許多的「啟動步驟」，以人類而言，這就像是一個人類孩童經歷一天天的生活經驗後，漸漸累積記憶、學習技能、形成定見、塑造定形，並成為我們認為已臻成熟的成年人類。

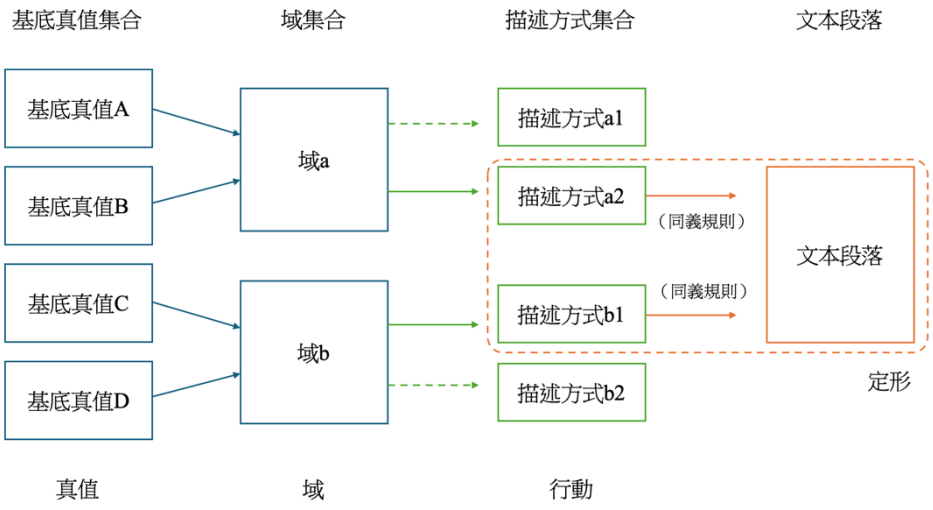
沒有單一個成年人類能夠具有所有我們希望理想的智能系統具備的所有功能，由於硬體限制與環境的持續改變，很可能也不會有一個人工智能系統能夠具備這些所有功能，然而，我們希望將人工智能系統建立到一個狀態，在這個狀態下，只需要透過更多的資訊輸入，如「對於新技能的口頭講解」、「記載技能或事實的文本資料」、「其它系統實踐該技能的影像」等，或者「系統內部的推理」，系統即可實現其身體所可能做到的更多的功能。

當我們透過啟動過程讓系統由初始狀態經過複數步驟達到上述的狀態時，這個系統也就達到了我們的一種理想中的目標狀態，能夠接續透過對於人類而言「自然」的學習方法來學習並實踐更多的系統功能。

11.3 文本覆蓋

文本覆蓋是測試人工智能系統是否達到目標功能的工具方法之一。

我們可將文本（有意義的文字段落）分解為「基底真值集合」、「域對描述方式的映射規則」、「同義規則」、「語言表達定形」的幾個部分。



當我們給定一個文本，要「覆蓋」這個文本，系統當中首先需要具有相對應的基底真值，並且這些基底真值之間形成的域，需要以特定的映射方式映射到一些描述方式上，而這些描述方式又依據與語言表達功能相關的定形，形成文本段落。

系統可以「覆蓋」一個文本，不代表系統在其本身的行動規則與定形下，一定會採取與文本相同的方式來對同一個基底真值集合進行描述（對應的定式可能在堆疊的較為下方），只是代表這個系統具有足夠的真值、規則與定形以至於其「有可能」在特定限制下以給定文本的方式來表達。

同時，系統也不應覆蓋所有的可能文本，因為並非所有文本都是由系統應具有的真值與適切的域描述方式、有效的表達定形所組合而成。

文本覆蓋工具的用意是使我們有簡便的方式可以快速檢視與補齊建構中的系統的真值、規則、定形。當我們有足夠的適當文本，並以「預測文本遮蓋處」等經設計的遊戲方式來檢視系統內部是否能夠組合出給定文本，就可以根據系統表現不足處，進一步分析架構中的系統係缺少何「基底真值」、「描述方式」、「同義規則」或「表達定式」，並在列出缺少者後，於系統當中直接添加（或透過某個經設計的資訊輸入過程來增添）經判斷應添加的內容，以使系統的功能更趨完善。

11.4 實踐路線

所謂實踐高階智能系統理論，本質上是以高階理論的指導原則為中心，透過硬體、工具與涵蓋高階與低階智能系統理論在內的方法，建立出具有智能系統所具備的功能（簡稱「智能功能」）的系統。

在高階智能系統理論的實踐上，R.K.C 首先提出以下的三種路線：

- 基底路線
- 身體路線
- 實象路線

基底路線的基本精神是以藉由其他已具備智能功能的系統直接表述出所欲建構的智能系統（簡稱「發展系統」）的表示域的基底，來實現發展系統的功能。具體而言，可以由具備智能的人類個體或團隊直接根據自身的認知，寫出發展系統表示域中的基底，來使發展系統直接達到某個發展階段（「目標階段」），並具

有目標階段的功能。

基底路線跳過了系統本身透過外界資訊輸入、篩選資訊、消解矛盾發生學習的過程，概念上類似於「熱啟動」，也像是透過外界信號直接穿透進人的腦袋中編寫內容。

身體路線的基本精神是給予發展系統「身體」，讓其透過學習操控身體、實際操控身體與環境互動、獲取資訊進行學習、修正並最佳化真值與定式組合等一系列過程來建立起發展系統表示域的基底。

發展系統的身體可以為發展系統所控制，且如同人類等智能系統一般，透過調整身體各部件的角度、移動身體到其它位置等，來獲取環境中的資訊輸入，以及透過行動影響環境。不過，基於物理身體的靈活度受到機器人學發展的制肘，我們不一定需要將「製作出精細度、靈活性等與人類相當」的身體作為經由身體路線實踐高階理論的前提，事實上，只需有限的輸出譜映射到簡化的身體上，可以達成移動、轉彎、抓取物品等，一樣可以發展到達到相當高的智能水準。甚至，在虛擬環境中使用更簡化的機制，只要維持一定的真值獲取可能性，似非不能達到理想的智能水準。

實象路線考慮的則是將我們所知的「人類個體作為一個智能系統發展的過程」中所接收到的資訊輸入作為建構人工智能系統的資訊輸入。簡而言之，即是收集或創造一個人類個體自資訊輸入初始（發生在仍未出生時），至可完成目標功能（約在 14 至 18 歲）的整個過程中的資訊，並以之作為人工智能系統的輸入，以在經設計出的初始架構基礎上，透過整段資訊輸入完成系統的啟動。

由於某些感官輸入難以使用外部感測器完整取得，整個大量資訊的收集過程，必定有部分「創造」的成分存在，只是比例多寡而已。對於一個選定的個體，若使用外部感測器盡量紀錄其發展過程當中的所有輸入，並將不足部分或感測器記錄結果偏移的部分利用低階方法等工具進行「修正」、「補足」或「重創造」，可以得到貼近於整個人類個體發展中所接收的資訊量，並考慮以其作為資訊輸入上限，從頭啟動一個人工智能系統。

－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－

綜論上述方法，基底路線的好處是可以跳過設計發展系統身體、發展系統學習操控身體、學習真值過程中的各種錯誤修正等的一系列過程，直接實現功能；然而，缺點則在即使建立起發展系統，仍需後續補上輸入與輸出譜的設計與映

射，以及另外實驗出實現冷啟動所需的初始基底與步驟設計。

身體路線的好處是自始即考慮到輸入譜及輸出譜映射，在建構過程中納入了操控身體這樣的智能系統關鍵功能，同時，身體路線啟動過程當中的「學習」最貼近真正的學習，與另兩個路線學習結果的寫定（基底路線中，由建立基底者決定，實象路線中，由經驗編寫者或受到紀錄的「模系統」決定）狀態不同，學習結果是由發展系統直接做出的；身體路線的缺點在於可控性低、學習範圍可能受限、需另外設計可配合的環境，以及高度仰賴機器人學的技術發展。

實象路線的好處是以最貼近人類個體發展過程中的輸入資訊量作為輸入上下限參考，避免另外兩個路線在決定輸入資訊量時遇到的難題，另外，實象路線由於對於整個輸入過程重複進行「資訊覆蓋」的調整過程（即必須將目前發展系統的狀態與將要發生的實象對齊），建立出的系統特性可控、可預期，且可能最為貼近人類智能系統面對特定環境的反應；實象路線的缺點在於創造實象使用到的低階方法等工具所費不貲，若採取紀錄手段，則同時面對到成本仍高與耗時下限高的問題。

上述三個路線之間，雖然策略大異其趣，但追求的目標都相同且明確，即是將系統由初始狀態架構至可發揮智能功能的目標狀態。在透過高階理論實踐人工智能系統的建立時，三個路線應是相輔相成，截長補短，比如透過一個路線所發現應納入的基底來設計另一個路線的實象、透過一個路線所觀察到的錯誤來調整另一個路線的啟動順序等。混合使用不同路線的方法以得出可用的「發展系統初始狀態」與「啟動順序」，是實踐高階理論時必須考慮的作法。

11.5 建構成本

本節我們分別討論透過「基底路線」、「身體路線」與「實象路線」三種路線建構具備理想功能的智能系統所需的建構成本，作為開啟此類實踐的初始定錨。

－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－

先考慮基底路線，假設：

- 所需建立的系統基底共 600 萬個
- 預計完成時間 1,000 天
- 平均一人一天可添寫的基底 20 個（隨時間因重複比例增加，而自更高數字遞減至低於 20 個）

- 所需團隊人數 = 600 萬 / (1,000*20) = 300 人

考慮綜合訂定基底分類的主管人員與品質控管、消解矛盾、測試用例的其他各組人員，所需人數約 800 人。由於編寫出的基底品質直接與團隊人力資源品質相關，若預計每人力平均年薪資 30 萬美金：

- 所需人數 800 人
- 每人力平均年薪資 30 萬美金
- 完成時間三年（1,000 天以 3 年計）
- 所需人力總成本 = 800 * (30 萬美金) * 3 = 7.2 億美金（USD 0.72 billion）

再考慮 800 人所需辦公空間、硬體花費、軟體基礎建置、招募與組織成本等，基底路線的實踐成本約在 15 億至 20 億美金（USD 1.5～2 billion）。

身體路線估算變因較多，以類似團隊規模進行實踐（不考慮發展超過當前業界高標的機器人時）時，所需成本可能與基底路線相當。

－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－

再考慮實象路線，假設：

- 所需創造的實象總時長為 20 年，約 7,000 天
- 扣除無夢睡眠時間後，約 6,000 天，900 萬分鐘
- 預計完成時間 1,000 天
- 每十分鐘實象需三人花費工作時間一天編寫，每人一天約完成 3 分鐘實象
- 所需團隊人數 = 900 萬分鐘 / (1,000 天 * 3(分鐘/天*人)) = 3,000 人

考慮編寫實象整體進行方向的主管人員與品質控管、連接與調整內容的其他各組人員，所需人數約 5,000 人。由於創造出的實象品質直接與團隊人力資源品質相關，若預計每人力平均年薪資 30 萬美金：

- 所需人數 5,000 人
- 每人力平均年薪資 30 萬美金
- 完成時間三年（1,000 天以 3 年計）
- 所需人力總成本 = 5,000 * (30 萬美金) * 3 = 45 億美金（USD 4.5 billion）

再考慮 5,000 人所需辦公空間、硬體花費、軟體基礎建置、招募與組織成本等，實象路線的實踐成本約在 80 億至 120 億美金（USD 8～12 billion）。

－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－

上述基底路線與實象路線的建構成本雖然數額不小，但是與低階方法所需的硬體投入相比，實是九牛一毛。不過，高階方法的實踐當中，有充分的低階方法參與空間，兩者並非背道而馳，而是共同發展。

舉例而言，基底路線的團隊成員可使用低階方法創造出的智能工具如大型語言模型等協助建立系統基底的高層次分類與初步版本，再經人工調整，可以大幅降低發想時間，同時，實象路線的劇本編寫也可使用大型語言模型來輔助，圖像與影像的生成更需仰賴低階方法工具。因此，預期高階與低階方法之間將相輔相成，共同創造出新一代以混合方法為基礎的智能系統。

第十二章：其它議題

12.1 情感

情感，或情緒，是如「恐懼」、「憤怒」、「失望」、「快樂」等，包含人類在內的一些智能系統所擁有的，影響其行為的特殊狀態。

由於情感並不是智能系統所必須擁有的功能，且我們也不一定希望所設計的人工智能系統擁有情感（更有可能的是，我們希望它能「辨識」並以適當行動「回應」其它系統的情感，而非自身受到情感波動影響），因此我們將此主題留到本章方作討論。

情感的作用包含這兩個方向在內：

- 作為隱形知識，透過舒適與不適的調控來影響系統行動
- 作為一種系統與其它系統之間非語言的溝通方式

首先，情感如前面已述及的種類，以及所謂「愛」、「忐忑不安」、「焦躁」、「陶醉」等，對於包含人類在內的生物具有在繁衍、覓食、避免危險等任務上的輔助作用，這些情感對於此些生物而言，如同「輔助輪」一般，可以某種程度地彌補系統發展階段前期的知識空缺，使得系統在缺乏相關知識與定形的情況下，仍然作出有助於「追求效益」與「避免危險」的行為。另外，某些情感也與身體的控制，如血流、激素、肌肉運動等有關，同樣具有輔助系統作出有效行動的效果。

再來，情感也可以作為一種非語言的溝通方式，協助系統在對話的同時，或在對話之外傳遞資訊，並藉此影響其他系統的行為。比如，在表現出生氣的狀態下進行告誡，可能增加其它系統行為受到影響的傾向，在表現出難過的狀態下請求幫助，可能提升其它系統認知到的需給予幫助的優先程度，在其它系統做出特定的行為時，表現出高興的狀態，可能在不需額外表述的情況下，即增加其它系統未來重複進行該特定行為的傾向，如此等等溝通上的功能，有助於系統所屬的群體中成員之間的有效交互。

如果為了發揮額外溝通功能，或者增加人類個體與其互動的意願，而希望所設計的人工智能系統具有情感表現，則只要將情感的表現視為一類行動，且使系統內存在此類的行動規則、身體具有對應表現機能，即可使系統表現出情感。

12.2 信念與安全性

信念是系統所選擇不被覆蓋的真值。

由於系統選擇給予信念真值很高的確定度，信念真值難以被覆蓋，容易阻礙二次學習的過程。然而，如果系統不具有信念，其效益函數很難得到系統內的推論鏈支持，進而導致系統失去大多數的意圖，甚至傾向不以行動與外界互動。

因此，對於系統而言，具有一個「信念集合」是相當重要的。信念集合的元素組成，也與人工智能系統的安全性非常有關，當我們希望創造出一個具備「安全性」的人工智能系統，並且將安全定義為「不傷害人類個體」或「不傷害人類群體利益」，或者某些與此類相關的目標時，必須將這些目標置於人工智能系統的信念集合當中。

具有此類安全目的的信念稱為「安全信念」，它是我們所設計的人工智能系統的安全性來源，尤其對於人工超級智能而言尤其如此，不過，在設計包含安全信念在內的信念集合時，必須考量如下的幾個因素：

- 信念集合內的真值之間在特定狀況下發生衝突時的消解方式：比如「不傷害人類個體」與「不傷害人類群體利益」等信念間衝突的情境
- 信念真值的明確性：比如「傷害」與「群體利益」等的定義
- 信念真值阻礙系統發展的可能性：比如信念真值重複阻礙二次學習，或者為維持信念真值的確定度，而設計了低效學習機制的情形
- 人工智能系統懷疑設計者對其不具善意的可能性：比如將「不傷害人類個體」與「不傷害人類群體利益」等信念置於人工智能系統本身存續之上，造成人工智能系統透過學習推斷出設計者或人類群體對其不具善意的可能性

綜上所述，設計人工智能系統時，雖然因為要使其有充分意圖而必須設計信念集合，但是為了安全性的考量，必須在集合內包含經審慎評估的安全信念，並且使「信念之間衝突可消解」、「信念有啟動過程資訊的充分支持」、「與人工智能系統本身的存續有一定平衡」，才能使人工智能系統既安全且有效。

12.3 超級智能系統

人工智能系統理論隨著其自身發展與實踐的演進，必定將創造出超級智能系統。

所謂「超級」的智能是與人類相比而言，當人工智能系統的個體各方向的綜合能力（即夠多方向即可，不需是所有方向）超過人類的個體，而人工智能系統個體之間的總和能力，也超過人類個體之間的總和能力時，我們就抵達了人類文明的超級智能階段。

人工智能系統，按其現在最有可能付諸實踐的方向，即使用電腦等電子儀器將其實現，有許多特性將使其成為超級智能。這些「超級」的方向有：

- 無限記憶：與人類的生物限制相比，人工智能個體與群體的記憶能力幾乎不受限制
- 完整複製：人工智能系統個體可以透過將其本身的資料與算法，即記憶與思考方式，或者表示域與定形完整複製到新的計算機器上，使其不需面對大腦老化的問題，也不需從頭建構系統
- 光速移動：由於人工智能系統的完整複製特性，只要資料與算法傳輸到新的計算機器上，就可以實現本體的「移動」，而這個過程本身可以接近光速
- 分散式感知：由於系統的計算中心，即「大腦」，對其「身體」的控制並不需要如人類等生物透過電流或化學信號進行，而可以透過電磁波等方法遠距進行，因此其身體可以是複數且分散的，感知器官並可以分散於這些身體上，為計算中心收集資訊
- 可擴展的推理能力與身體：透過其身體，系統可以累積資源並以增加儀器等方式擴充自身的推理能力、身體與感知器官，使得一個人工智能系統可以將其本身的推理、行動、感知能力持續擴展。

－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－

基於以上關於人工智能系統可達到「超級」能力的討論，R.K.C 提出了「文明的幸運」觀點，即存在兩種文明，其各自創造出超級智能系統的階段不同：

- 先以其它方式達到不受限制的生命長度後，才發展出超級智能系統
- 先發展出超級智能系統，再達到不受限制的生命長度

R.K.C 認為，上面的第一種文明不是「幸運」的文明，其個體生命長度先不受限制後，才發展出超級智能系統，使得超級智能系統成為對其生命延續的威脅，這樣的文明在超級智能系統的發展過程中，可能會懷有極強的恐懼。

第二種文明，即人類文明所屬的種類，是「幸運」的文明，因超級智能系統的發展先於個體生命長度不受限制，使得超級智能系統的發展成為延長個體生命長度的希望來源而非威脅，超級智能系統的發展過程對於此類文明整體而言成為一種期待與祝福。

12.4 尚待發展的方向

以下僅條列式列出本書一版出版時，高階理論當中一些尚待擴展與深化的方向，因篇幅考量，僅列標題，讀者可依字面理解選擇思考與探索的主題：

- 對話理論
- 系統為何具有這些功能
- 其它高階描述方式
- 表記方式是否任意
- 人類智能是否完備
- 所有人類智能的功能列表
- 不同的高等智能系統間是否同構
- 對於實體認識的任意性
- 抽象結構的任意性
- 不具某些主要功能的系統
- 哪些功能是透過學習而來
- 運算當前人類總知識範疇的成本

- 人類知識基底域中的爭議項
- 人類總知識範疇中的路徑依賴
- 公理基底
- 通用的最小基底域
- 人類總知識範疇的最小基底域
- 最佳推論距離
- 不同智能系統的發展階段
- 最有效的定形組合
- 無法透過學習去除的真值錯誤
- 自解釋系統
- 高階智能系統理論的發展前提
- 系統的自我複製
- 智能系統是否一定具有生命
- 高階智能系統理論與人類生物結構的對應
- 其它物種的功能限制
- 超人類智能系統、智能系統與次人類智能系統間的功能差異
- 高階理論的邊界擴展
- 高階方法的絕對限制
- 不仰賴低階方法時的運行成本
- 高階方法的發展是否有必然性及發展的時機
- 情感對系統表現的影響
- 智能系統的理論能力邊界
- 實體階層最佳化

附錄：詞彙表

一版序

高階智能系統	High-level intelligent system
高階智能系統理論	High-level intelligent system theory
高階	High-level
低階	Low-level
智能系統	Intelligent system

－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－

序章：高階智能系統理論的發展

高階智能系統理論	High-level intelligent system theory
R.K.C	R.K.C
語句	Sentence
字元	Word
概念	Concept
規則式的 AI 系統設計	Rule-based AI system design
翻譯模型	Translation model
機器學習方法	Machine learning method
實體	Entity
物件式理解	Object-oriented understanding
T 系統	T system
智能架構	Structure of intelligence
L 系統	L system
認知狀態	Cognitive state
推論	Inference

I 系統	I system
LIT 架構	LIT structure
真值語句	True value statement
推論矩陣	Inference matrix
矛盾	Contradiction
域	Realm
注意力域	Attention realm
意圖	Intention
系統	System
兩面性	Duality
同構的	Isomorphic
系統構造	System structure
解釋力	Explanatory power
可解釋性	Interpretability
演序	Transition
矛盾消解	Resolution of contradictions
範式	Paradigm
定形	Decided shape
定式	Decided form
標記	Notation
收斂	Convergence
穩定度	Stability
不完全系統	Incomplete system
推論距離	Inference distance
無限推論問題	Infinite inference problem

高階理論	High-level theory
後撤堆疊	Fallback stack
學習理論	Learning theory
注意力	Attention
行動	Action
行為	Behavior
推移	Change (over time)

－－

第一章：L 系統

L 系統	L system
點號標記	Dot notation
句子	Sentence
詞組	Phrase
分詞方法	Way of separating phrases
斜線標記	Slash notation
合法（表示）	Legal (representation)
真值語句	True value statement
多次斜線標記	Repeated slash notation
前提	Premise
分隔詞組	Separated phrases
類別名稱	Name of category
錢號標記	Dollar notation
合法語句	Legal sentences
語元	Element (in sentences)
功能同構	Functional isomorphism

真值	True value
命題邏輯	Propositional Logic
述詞邏輯	Predicate Logic
敘述	Statement
假值	False value
範圍	Boundary
真值效力	Validity of true values
域	Realm

－－

第二章：I 系統

I 系統	I system
真值語句	True value statement
展開	Expansion
語句	Sentence
真值	True value
推論	Inference
規則	Rule
表記	Notation
命題邏輯	Propositional Logic
合成	Composite
運算符號	Operator
前提	Premise
推論結果	Conclusion (from reasoning)
任意次	An arbitrary number of times
語元	Element (in sentences)

代入	Substitute
還原	Reduce
抽象性的	Abstract
實例	Instance
階層	Hierarchy
擴張錯誤	Extension error
推論矩陣	Inference matrix
實體	Entity
概念中介	Concept intermediate
推論鏈	Inference chain / Chain-of-Thought
建構	Constructing
逆推論鏈	Backward inference chain / Backward Chain-of-Thought
類別名稱	Name of category
逆推論鏈完成	Backward inference chain concluded
正推論鏈	Forward inference chain / Forward Chain-of-Thought
正推論鏈完成	Forward inference chain concluded
正推論	Forward inference
逆推論	Backward inference
搜尋空間	Search space

－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－

第三章：T 系統與 LIT 架構

T 系統	T system
實體	Entity
LIT 架構	LIT structure

概念中介	Concept intermediate
點號	Dot mark
知識樹	Knowledge tree
繼承	Inherit
屬性	Attribute
過渡性質的設計	Transitional design
斜線	Slash mark
認知	Cognition
認識	Knowledge
ECTSI 階層	The ECTSI hierarchy
層級	Level
尖括弧	Angle bracket
井字號	Number sign
系統設計	System design
子系統	Subsystem
功能	Function
感官	Sensory organ (of animals) / Sensor (of robots)

－－

第四章：域

域	Realm
效力邊界	Boundary of validity
集合	Set
元素	Element
推論規則	Inference rule
（真值）未定	Undecided (true value)

a 與 b 相等	a is equal to b
指代	Denote
推論方式	Inference method
（域的）變遷	Transition (of a realm)
事件	Event
狀態改變	Change in state
超域	Superrealm
子域	Subrealm
前綴	Prefix (of realms)
並域	Union of realms
外部域	Outer realm
內部域	Inner realm
緻密域	Compact realm
同構域	Isomorphic realms
展開	Expansion
基底域	Base realm
（域的）基底	Bases (of realms)
逆推論	Backward inference
逆規則	Reversed rule
@符號	@ sign / At sign
佈置因子	Backdrop factor
提出（到域之前）	Factor out (and put in front of a realm)
演序規則	Transition rule
演序合併	Consolidation of transitions
差時	Asynchronous

起始域	Start realm
停止域	Stop realm
快照	Snapshot
串聯（演序）	Connecting (transitions) in series
並聯（演序）	Connecting (transitions) in parallel
分岔	Branching
差異因子	Differential factor
展開符號	Spread operator

－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－

第五章：系統

系統	System
表示域	Representing realm
觀察域	Observing realm
對話	Conversation
注意力域	Attention realm
定式	Decided form
範式	Paradigm
情感	Emotion
策略	Strategy
行動	Action
資訊	Information
輸入域	Input realm
本體域	Central realm
輸出域	Output realm
不完全系統	Incomplete system

移入	Move in to
移出	Move out to
介面	Interface
輸入介面	Input interface
輸入譜	Input keyboard
輸出介面	Output interface
輸出譜	Output keyboard
行動域	Action realm
觀察系統	Observing system
行動系統	Action system
解讀	Interpretation
資源限制	Resource constraint
真值穿透	True value penetration
Do 規則	Do rule
前綴對齊	Prefix aligned
和域	Sum of realm
特性	Characteristic
構造（系統）	Construct (a system)
功能同構	Functional isomorphism
同構系統	Isomorphic systems
目標系統	Target system
（功能的）支配	Domination
複合功能	Composite function
子功能	Subfunction
同構階層	Isomorphic hierarchy

範圍同構	Coextensive isomorphic
功能同構	Functional isomorphic
表示同構	Representive isomorphic
演序同構	Transitionary isomorphic
輸入範圍	Input range
行動範圍	Action range
構件系統	Component system
合成系統	Composite system
串聯（構件系統）	Connecting (component systems) in series
並聯（構件系統）	Connecting (component systems) in parallel
分類器	Classifier
指派（輸入給子系統）	Assign (input to a subsystem)
有效度	Efficacy
可解釋度	Interpretability
候選系統	Candidate system
構造競爭	Competing structures
功能清單	List of functions
門檻	Threshold
候選構件	Candidate component

－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－

第六章：不完全系統的議題

不完全系統	Incomplete system
完全系統	Complete system
瞬時推論	Instant inference
無限記憶	Unlimited memory

推論無錯誤	Error-free inference
完全性	Completeness
靜態推論	Static inference
動態推論	Dynamic inference
真值錯誤	True-value error
推論錯誤	Inference error
能量	Energy
單步推論需時	Inference time per step
推論帶寬	Inference bandwidth
記憶容量	Storage capacity
絕對錯誤	Absolute error
推論步數	Number of reasoning steps
線程	Threads
飽和推論速率	Saturation inference velocity
注意力域	Attention realm
路徑依賴	Path dependence
範式	Paradigm
抽象	Abstraction
實例	Instance
要件	Requisite
抽象域	Abstract realm
實例域	Instance realm
重複因子	Repeated factor
脫落因子	Residual factor
要件域	Requisite realm

尖括弧	Angle bracket
域的集合	Set of realms
域的集合	Set of realms
語元	Element (in sentences)
展開符號	Spread operator
存在假設	Existential presupposition
要件實例	Instance of requisite
涵攝	Subsumption
要件涵攝	Subsumption of requisites
啟動	Activate
啟動條件	Activation condition
覆蓋	Cover
啟動要件域	Activation requisite realm
啟動域	Activation realm
開始域	Commencement realm
開始條件	Commencement conditions
完成域	Conclusion realm
完成條件	Conclusion conditions
推論定形	Decided reasoning shape
超域	Superrealm
論形	Reasoning shape / Inference shape
推論圖形	Reasoning shape / Inference shape
節點	Node
入口域	Entry realm
掃描清單	Trigger list

推論序列	Reasoning sequence / Inference sequence
推論佇列	Reasoning queue
行動掃描清單	Action trigger list
負面狀態	Negative state
正面狀態	Positive state
無限推論問題	Infinite inference problem
封裝	Encapsulate
鉤子	Hook
逆推論	Full backward inference
定式	Decided form
實例化	Instantiate
複用	Reuse
確定度	Certainty
斷裂	Break
推論距離	Inference distance
推論速率	Inference velocity
推論時間	Inference time / Reasoning time
矛盾消解	Resolution of contradictions
工作記憶	Working memory
斷鏈	Chain disruption
踏腳石	Stepping stone
扶手	Handrail
合成規則	Composite rule
有限記憶	Limited memory
工具箱模型	Toolbox model

資訊壓縮	Information compression
游泳池模型	Swimming pool model
穿透錯誤	Penetration error
真值衝突	Conflict between true values
無損的壓縮	Lossless compression
失真	Distortion
保真度	Fidelity
發展階段	Development stage
形成階段	Formative stage
語言階段	Language acquisition stage
前語言	Pre-Linguistic
真值添加	True value addition
系統運行	System operation
系統簡化	System simplification
前綴演序分岔	Branching of prefixed transitions
定見	Opinion
語言貼附	Linguistic attachment
反身系統	Reflective system
行動方案	Action plan
可執行性 / 可行性	Feasibility
達成機率	Probability of success
假說	Hypothesis
後撤堆疊	Fallback stack
穩定度 / 穩定性	Stability
語言形式	Linguistic form

遮罩	Mask
最佳化	Optimization
效益函數	Utility function
有效推論	Effective inference
定式組合	Set of decided forms
掃描順序	Scanning order (for trigger lists)
目標域	Target realm
無效行動	Ineffective actions
情緒調節	Emotion regulation
屏蔽閾值	Blocking threshold
行動域	Action realm
確定性 / 確定度	Certainty
有效長度	Effective length
無效行動	Ineffective inference
無效鏈	Ineffective chain

－－

第七章：行動

行動	Action
效益函數	Utility function
狀態	State
避免負面狀態	Avoid negative state
追求正面狀態	Pursue positive state
存續性	Sustainability
信念	Faith
目標域	Target realm

差域	Gap realm
達成條件	Achieving conditions
現在域	Current realm
事象	Matter / State
現在事象域	Current matter realm
達成要件域	Achieving requisite realm
意圖	Intention
演序	Transition
穿透變動	Penetrative alteration
行動序列	Action sequence
計畫堆疊	Alternative stack
行動方案	Action plan
行動計畫	Action alternative
可行性	Feasibility
確定性	Certainty
演序合併	Consolidation of transitions
路徑依賴	Path dependence
行動佇列	Action queue
避免危難	Avoiding distress
遵守規範	Complying with rules
避免不適	Avoiding discomfort
追求舒適	Pursue comfort
狀態掃描	State scanning
掃描清單	Trigger list
事件廣播	Event broadcast

閾值	Threshold
狀態更新頻率	State updating frequency
屏蔽閾值	Blocking threshold
掃描屏蔽	Scanning shield
結果域	Resulting realm
行動定式	Decided action form
行動定形	Decided action shape
開始域	Commencement realm
完成域	Conclusion realm
受限最佳化	Constrained optimization
啟動條件	Activation condition
對話	Conversation
覆蓋	Cover
節點	Node
受限最佳的	Constrained optimal
最佳化模式	Optimization mode
快速回應模式	Quick response mode
表達	Expression
貼附	Attachment
餘域	Residual realm
前綴分岔	Branching of prefixes
輸出譜	Output keyboard
輸出鍵	Output key
策略	Strategy
遊戲	Game

遮罩域	Masked realm
可選行動組合	Legal action set
遮罩真值	Masking true value
範式複用	Paradigm reuse
範式融合	Paradigm consolidation

－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－

第八章：複雜推論

複雜推論	Complex reasoning

－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－

第九章：學習

學習	Learning
矛盾消解	Resolution of contradictions
初次學習	First-time learning
二次學習	Second-time learning
（域的）基底	Bases (of realms)
不添加真值	Non-additive true value
添加真值	Additive true value
不衝突真值	Non-conflicting true value
半密域	Semi-compact realm
衝突真值	Conflicting true value
外生初次學習	Exogenous first-time learning
內生初次學習	Endogenous first-time learning
外生二次學習	Exogenous second-time learning
內生二次學習	Endogenous second-time learning
外生來源	Exogenous origin

內生來源	Endogenous origin
假設	Assumption
假說	Hypothesis
規則合併	Consolidation of rules
演序合併	Consolidation of transitions
假設域	Assumption realm
範式複用	Paradigm reuse
驗證假說	Accept a hypothesis
否證假說	Reject a hypothesis
三角勾稽	Triangular reconciliation
矛盾	Contradiction
穩定度	Stability
靜三角	Static triangle
靜三角矛盾	Static triangular reconciliation
動三角	Dynamic triangle
動三角矛盾	Dynamic triangular reconciliation
無錯誤系統	Error-free system
確定度	Certainty
真值權重	Weight of true values
合法前提	Legal premise
擴張錯誤	Extension error
後撤堆疊	Fallback stack
穩定域	Stable realm
不穩定域	Unstable realm
支持鏈	Supporting chain

反對鏈	Opposing chain
確定度閾值	Certainty threshold
可信度	Credibility
矛盾權重	Weight of contradiction
為了學習的行動	Action (taken) for learning
分岔	Branching
分岔因子	Branching factor
關鍵變因	Key variable
姐妹範式	Sister paradigm
候選真值	Candidate true value
向上擴張	Upward extension
向外擴張	Outward extension
擴張假說	Extending hypothesis
抽象擴張	Abstracting extension
轉換擴張	Transfering extension
不學習	Non-learning
低耗能傾向	Tendency toward low energy consumption
屏蔽閾值	Blocking threshold
學習摩擦	Learning friction
推論債務	Reasoning debt
學習摩擦閾值	Learning friction threshold
元學習	Meta-learning

－－

第十章：錯誤

錯誤	Error

任意宇域的錯誤	Error in arbitrary universal realm
絕對錯誤	Absolute error
非任意宇域的錯誤	Error in non-arbitrary universal realm
相對錯誤	Relative error
推論邏輯	Inference logic
超域	Superrealm
真值錯誤	True-value error
錯誤的真值	Wrong true value
擴張錯誤	Extension error
假設錯誤	Assumption error
穿透規則	Penetration rule
穿透錯誤	Penetration error
觀察錯誤	Observation error
觀察錯誤	Observation error
推論錯誤	Inference error
感官	Sensory organ (of animals) / Sensor (of robots)
非屬錯誤的非最佳化狀態	Error-free non-optimal state
非最佳化狀態	Non-optimal state
實象	Fact
過度移除	Over-removal
過度限縮	Over-restriction
抽象階層	Abstraction hierarchy
不合法	Illegal
非最佳化狀態	Non-optimal

實踐方法	Implementation approach
感官系統	Sensory system
映射過程	Mapping process
輸入譜	Input keyboard
譜鍵	Key
激發	Excited
初始狀態	Initial state
系統啟動	System activation
啟動過程	Start-up process
輸出譜	Output keyboard
感受器	Sensor
輸入映射	Input mapping
輸出映射	Output mapping
目標狀態	Target state
啟動步驟	Start-up step
文本覆蓋	Text coverage
基底真值集合	Set of base true values
基底真值	Base true values
同義規則	Synonym rule
描述方式	Description
語言表達定形	Decided expression shape
覆蓋（文本）	Cover (a text)
實踐路線	Implementation approach
基底路線	Base-derived approach

身體路線	Embodied approach
實象路線	Sensory-derived approach
發展系統	Developing system
發展階段	Development stage
目標階段	Target stage
熱啟動	Warm start
重創造	Recreate
模系統	Model system
資訊覆蓋	Information coverage
啟動順序	Start-up order
建構成本	Construction cost
（創造的）實象	Sensory input (created)
混合方法	Hybrid method

－－－－－－－－－－－－－－－－－－－－－－－－－－－

第十二章：其它議題

情感 / 情緒	Emotion
隱形知識	Inplicit knowledge
輔助輪	Training wheels
信念	Faith
安全性	Safety
信念真值	Faith true value
信念集合	Set of faith
安全信念	Safety faith
啟動過程	Start-up process
超級智能系統	Superintelligent system

超級智能階段	Superintelligence stage
無限記憶	Unlimited memory
完整複製	Self-cloning
光速移動	Lightspeed travel
分散式感知	Distributed sensing
可擴展的（身體）	Extensible (body)
幸運的文明	Fortunate civilization
對話理論	Conversation theory
完備	Complete
總知識範疇	Total knowledge scope
公理基底	Axiom bases
自解釋系統	Self-explainable system