## Articut/Loki 的運作細節：以全稱語意為例

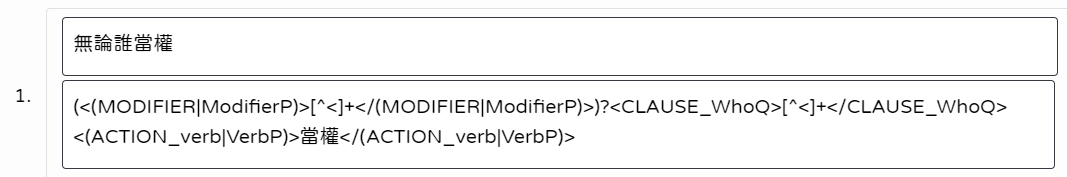
以下用全稱語意 (程式中註記為universal\_use) 為例說明運用 Articut/Loki 處理中文 *wh* 詞詮釋的流程。首先，我們收集中研院平衡語料庫中 (<https://asbc.iis.sinica.edu.tw/>) 所有包含「誰」的語料，總計2979 筆。經語料預處理過後抽取出含有「誰」的語料為 2695 筆。此處給定之例句為「無論誰當權」。系統在Articut將其表層解析為 (49)，其結構在(50) 以 regex 表示式呈現。而在系統後台，則將該例句儲存為 (51) 以符合 lambda 邏輯式呈現的真值條件。

1. ) 無論/誰/當權
2. ) <MODIFIER>無論</MODIFIER><CLAUSE\_WhoQ>誰</CLAUSE\_WhoQ>

<ACTION\_verb> 當權</ACTION\_verb>

1. ) ∀x (無論 (CLAUSE\_WhoQ(x)🡪 當權(x))

而Loki同樣使用 regex 表示式的概念，而使同一結構的句型能夠被判讀。系統初始判斷此句的結構如圖八所示:



圖八：「無論誰當權」的結構呈現與語意解式邏輯式

其中「|」符號意指 "or", 以圖八出現的 "<(MODIFIER|ModifierP)>" 為例，表句中可出現MODIFIER 或是 ModifierP。

而 「[^<]+」 符號表任意字詞，以 <(MODIFIER|ModifierP)>[^<]+

</(MODIFIER|ModifierP)> 為例，「[^<]+」 置於詞性標記中間，意即只要句中出現符合MODIFIER 或是 ModifierP 的任意字詞，此句型都會將其判讀。

再來是將帶有詞性標記的字串置於括弧中，並於後括弧處附加「?」符號，以 (<(MODIFIER|ModifierP)>[^<]+</(MODIFIER|ModifierP)>)? 為例，意指整段帶有詞性標記的字串為可有可無。以此句為例，也就是句子中的MODIFIER或是ModifierP不一定需要出現。

同時，為使「無論誰當權」此句結構能夠符合本研究主題，需要限縮此 regex 表示式，並同時達到正確判讀同一類的句型之目的。需要修改 regex 表示式的內容，加上原句「無論誰當權」用詞的同義詞，並且刪減可有可無的結構，形成最終圖九的 regex 表示式



圖九：限縮「無論誰當權」的結構呈現與語意解式邏輯式

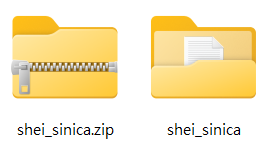
其中 (<FUNC\_inter>不論</FUNC\_inter>|<MODIFIER>無論</MODIFIER>)，「不論」和「無論」兩個帶詞性標記字串，中間的 「|」 符號意指 "or"，也就是句中可以出現「不論」或者「無論」，而此段 regex 表示式中加上的一段 (<AUX>是</AUX>)?，意指助詞「是」不一定需要出現。而原本的 <CLAUSE\_WhoQ>[^<]+</CLAUSE\_WhoQ> 意指所有符合 <CLAUSE\_WhoQ> 詞性的任意字詞都會被判讀，在這裡為符合本研究關於中文 *wh* 詞「誰」的主題，將詞性 <CLAUSE\_WhoQ> 之間由任意字詞 「[^<]+」 改為僅限字串「誰」。最後，動詞「當權」並不是本句型必要出現的動詞，故在此regex表示式將其刪減。

給定例句以後，系統自動設定完成表層的結構呈現以及深層的語意解釋邏輯式。此時點擊頁面左上角的 Home 圖示，回到最上層，選擇下載系統自動生成以 Python 程式語言執行的範本。



圖十：下載「無論誰當權」Python程式語言執行範本

範本下載後為一 zip 格式的檔案如圖十一，將之解壓縮得到一與專案名稱同名的 shei\_sinica目錄：



圖十一：解壓縮「無論誰當權」Python程式語言執行範本

進入shei\_sinica目錄，可見一意圖設定目錄、一帳號資訊檔及一主程式。

以句子「無論誰當權」為例，對於此句的處理機制，系統在兩種模式下分別執行。聊天機器人模式 (60 行的 CHATBOT\_MODE) 和非聊天機器人模式 (62 行的 else:)，系統更進一步處理該句子。此研究非重視回覆多樣性之聊天機器人，故使用非聊天機器人模式 (CHATBOT\_MODE = False)

首先，以句子「無論誰當權」為例，在 63 行將此句子加入通用意圖列表 "universal" 中，以表達系統辨識到該句符合此語意的結構。其次，系統以前文示例之修正過的結構 (<FUNC\_inter>不論</FUNC\_inter>|<MODIFIER>無論</MODIFIER>)(<AUX>是</AUX>)?<CLAUSE\_WhoQ>誰</CLAUSE\_WhoQ> 為依據，更新 "intent" 列表，將 "c3" 加入其中。這裡 "c3" 為自定義編號，"c" 表徵全稱語意，"3" 表示全稱語義中的其中一個句子的結構 (pattern)。並且 64 行出現的 "a1", "a5" 和 "a9" 也是自定義編號，"a" 表示疑問語意，"a1", "a5" 和 "a9" 各對應到不同包含強疑問詞的句子結構，分別為以下所示:

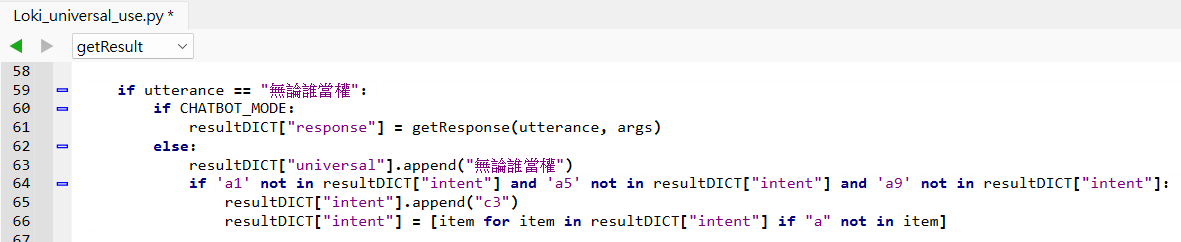
1. ) 強疑問詞結構：

a1: 誰+sentence-final interrogative particle

a5: 還有+誰

a9: 誰+ conjunction + noun or 誰

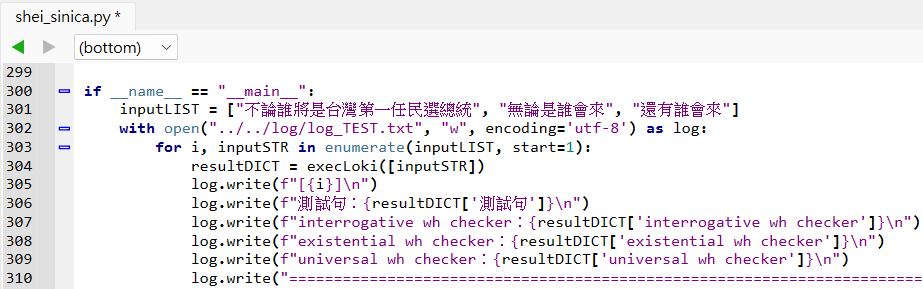
(52) 的結構被列為強疑問詞，也就是當以上結構和全稱語義的核可詞同時出現在句子時，檢查器會將其判斷為疑問語意，如圖十二的第 64 和 66 行所示。本研究預計將在此區塊將生成語言學理論實作為程式邏輯以進行大量測試理論完備性。



圖十二：「無論誰當權」Python程式語言執行範本

接著在主程式中填入待測試用的句子，此處以 (53) 三句為例做說明，測試結果如圖十三所示[[1]](#footnote-1)。系統執行後，如 304 行起所示，在 301 行將每個測試的句子逐一交由 execLoki() 執行，並在302行中名為log\_TEST.txt的檔案中寫入進行測驗結果，計算以判讀是否符合前述的已預先設定之結構。

1. ) 測試句：
   1. 不論誰將是台灣第一任民選總統
   2. 無論是誰會來
   3. 還有誰會來



圖十二：(53) 測試句檢驗截圖

再以「不論誰將是台灣第一任民選總統」為例，其結構在圖十三以regex表示式呈現。該句表層結構包含給定例句「無論誰當權」的表層結構。圖十三中的「將」、「台灣」、「第一任」、「民選」和「總統」以句子結構的判讀需求來說，皆為可有可無之「修飾詞」和可替換而不影響結構之「名詞」，故不影響表層結構之判斷。



圖十三：「不論誰將是台灣第一任民選總統」的結構呈現

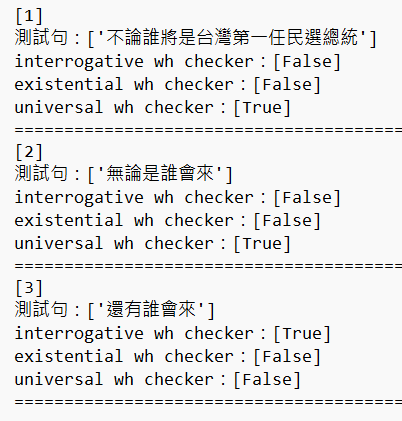
當表層結構中含有預給定的例句結構後，系統接著進入深層結構之轉譯。以「不論誰將是台灣第一任民選總統」為例，將其表層結構相符指定例句之部份轉為深層結構如(54)，由於系統將「修飾詞」和可替換而不影響結構之「名詞」判讀為皆為可有可無的函式，故可將其簡化為(55)。

1. ) ∀x∃y 不論(CLAUSE\_Who(x)) & 總統(y) -> 將(是(y, x) & 民選(y) & 台灣第一任(y))
2. ) ∀x (不論(CLAUSE\_WhoQ(x)🡪 是(x))

(55) 除了助動詞「是」之外，與前述之(51)相容。加上在圖九做的更動調整，在「不論」後加入可有可無的助動詞「是」，並且不一定需要動詞的出現，以符合更多相似結構，成為 (56)，以表示該輸入句「不論誰將是台灣第一任民選總統」結構與指定的 utterance「無論誰當權」相符。

1. ) if utterance == "無論誰當權":

如此反覆將每個要測驗的句子輸入程式中，即可如圖十四所示得出 (53) 之中每個句子依理論邏輯的計算結果是否相符於人類直觀的判斷結果 (如「是否呈現全稱語意」)，並依此檢驗理論邏輯上之完備性。



圖十四：log\_TEST.txt檔案內(53) 測試句的檢驗結果

1. 目前進行的小規模實驗中，皆可正確判讀「疑問、存在、全稱」三種語意，如圖十二所示，即已正確判讀三個句子裡的前兩個具有存在語意，而第三個沒有。針對中研院平衡語料庫中2979 筆含有「誰」的語料，本計畫已經建立了共 42 種語句模式 (疑問 24 種、存在 7 種、全稱 11 種)，正確的語意判斷比例為：疑問語意98%、存在語意 99%、全稱語義 99%；錯誤案例包括：某些不合法句、古文語句、句尾有「呀」的語句，本計劃將繼續針對這三類錯誤案例撰寫程式並擴大測驗範圍，以提供整體量化的正確率資料。 [↑](#footnote-ref-1)