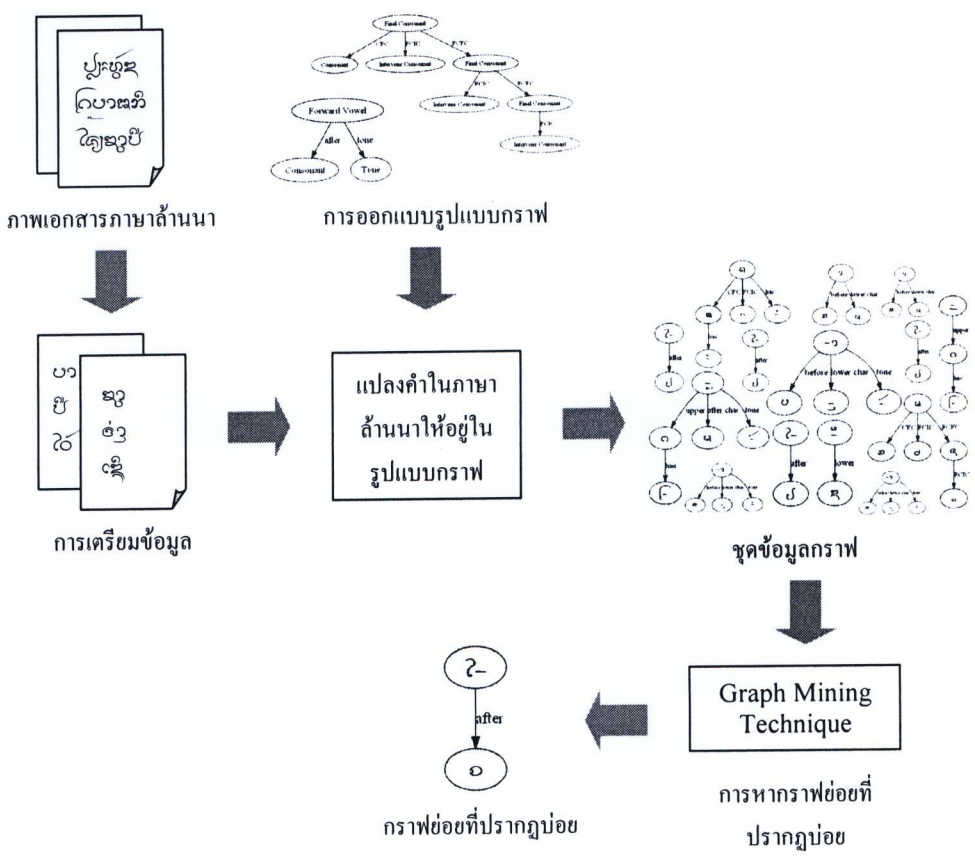
บทที่ 3
วิธีการดำเนินการวิจัย

การวิเคราะห์โครงสร้างคำภาษาล้านนาโดยใช้เทคนิคการทำเหมืองกราฟ มีขั้นตอนการดำเนินการวิจัยดังรูปที่ 3.1 ในการดำเนินการวิจัยประกอบไปด้วยขั้นตอนทั้งหมด 4 ขั้นตอน คือ ขั้นตอนการออกแบบรูปแบบกราฟที่เป็นตัวแทนของโครงสร้างคำภาษาล้านนา ขั้นตอนในการเตรียมชุดข้อมูลที่ใช้ในการวิจัย ขั้นตอนของการแปลงคำภาษาล้านนาให้อยู่ในรูปแบบของกราฟ และขั้นตอนของการวิเคราะห์โครงสร้างคำ ซึ่งเนื้อหาในบทนี้จะกล่าวถึงรายละเอียดเกี่ยวกับขั้นตอนในการออกแบบรูปแบบกราฟที่เป็นตัวแทนของโครงสร้างคำภาษาล้านนา และขั้นตอนของการแปลงคำภาษาล้านนาให้อยู่ในรูปแบบของกราฟ ส่วนรายละเอียดเกี่ยวกับการเตรียมชุดข้อมูลที่ใช้ในการวิจัย และการวิเคราะห์คำเพื่อหาโครงสร้างคำที่ปรากฏบ่อยนั้นจะกล่าวในบทถัดไป

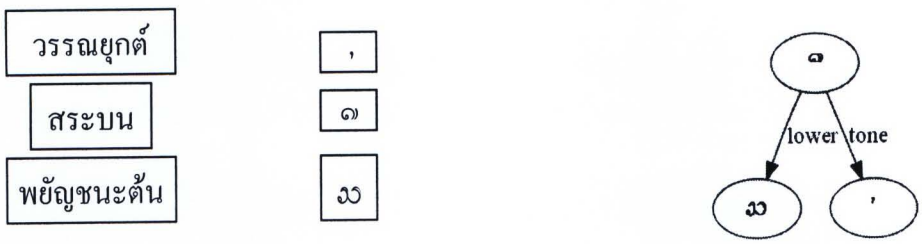


รูปที่ 3.1 ขั้นตอนในการดำเนินการวิจัย

3.1 การออกแบบกราฟที่เป็นตัวแทนของโครงสร้างคำภาษาล้านนา

ในการออกแบบกราฟจะแบ่งกราฟตามประเภทของคำทั้ง 5 ประเภท คือ กราฟของคำผสมสระ กราฟของคำที่มีตัวสะกดในตำแหน่งด้านล่างพยัญชนะต้น กราฟของคำที่มีตัวสะกดในตำแหน่งด้านหลังพยัญชนะต้น กราฟของคำที่มีพยัญชนะซ้อน และกราฟของคำที่มีตัวไหล จากการวิเคราะห์โครงสร้างคำของภาษาล้านนา สามารถแปลงคำภาษาล้านนาให้อยู่ในรูปแบบของกราฟได้ดังตัวอย่างต่อไปนี้

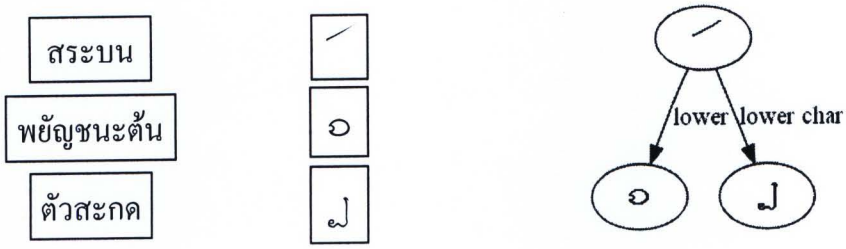
1. คำผสมสระ เช่น ᨾᩱ (สี่) มีโครงสร้างคำภาษาล้านนาตามรูปที่ 3.2(ก) สามารถแปลงโครงสร้างคำดังกล่าวให้อยู่ในรูปแบบของกราฟได้ดังรูปที่ 3.2(ข)



(ก) (ข)

รูปที่ 3.2 ตัวอย่างโครงสร้างคำและรูปแบบกราฟของคำผสมสระ

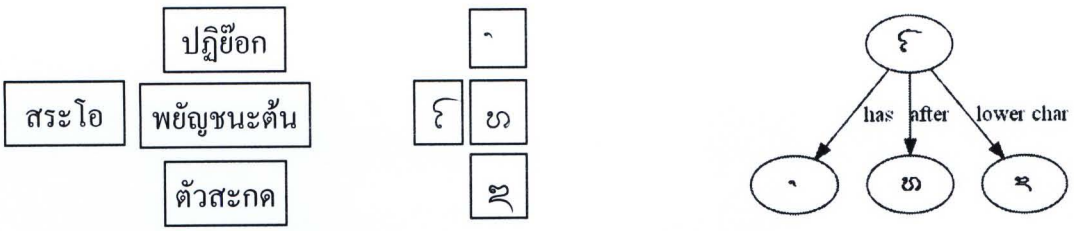
2. คำที่มีตัวสะกดวางไว้ด้านล่างพยัญชนะต้น เช่น ᨩᩢ (จับ) มีโครงสร้างคำตามรูปที่ 3.3(ก) ซึ่งสามารถแปลงโครงสร้างคำดังกล่าวให้อยู่ในรูปแบบของกราฟได้ดังรูปที่ 3.3(ข)



(ก) (ข)

รูปที่ 3.3 ตัวอย่างโครงสร้างคำและรูปแบบกราฟของคำที่มีตัวสะกดวางไว้ด้านล่างพยัญชนะต้น

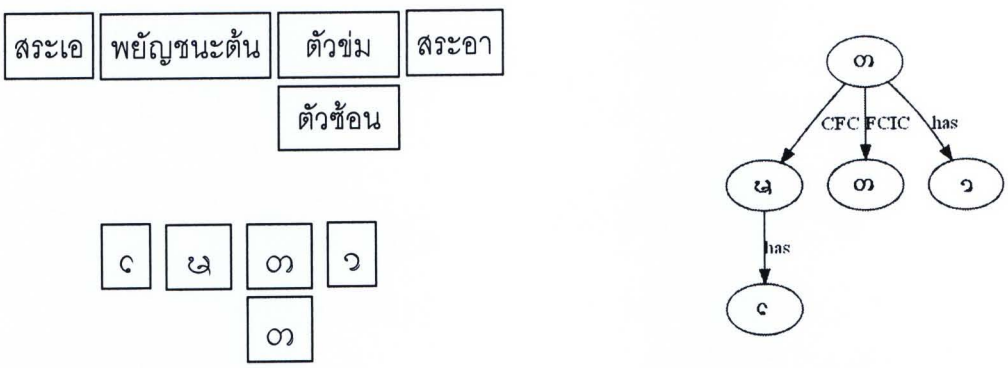
3. คำที่มีตัวสะกดวางไว้ด้านหลังพยัญชนะต้น เช่น ᩉᩮᩬ (โหด) มีโครงสร้างคำตามรูปที่ 3.4 (ก) ซึ่งสามารถแปลงโครงสร้างคำดังกล่าวให้อยู่ในรูปแบบของกราฟได้ดังรูปที่ 3.4(ข)



(ก) (ข)

รูปที่ 3.4 ตัวอย่างโครงสร้างคำและรูปแบบกราฟของคำที่มีตัวสะกดวางไว้ด้านหลังพยัญชนะต้น

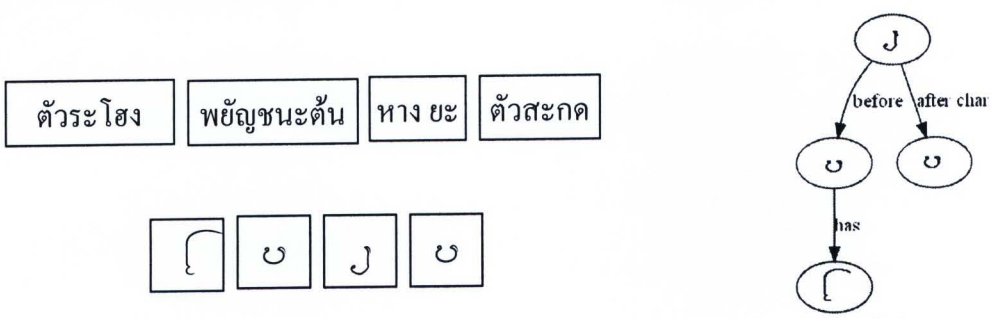
4. คำที่มีพยัญชนะซ้อน เช่น ๔๘๓๖ (เมตตา) มีโครงสร้างคำตามรูปที่ 3.5(ก) ซึ่งสามารถแปลงโครงสร้างคำดังกล่าวให้อยู่ในรูปแบบของกราฟได้ดังรูปที่ 3.5(ข)



(ก) (ข)

รูปที่ 3.5 ตัวอย่างโครงสร้างคำและรูปแบบกราฟของคำที่มีพยัญชนะซ้อน

5. คำที่มีตัวไหล เช่น เปรียบ (เปรียบ) มีโครงสร้างคำตามรูปที่ 3.6(ก) ซึ่งสามารถแปลงโครงสร้างคำดังกล่าวให้อยู่ในรูปแบบของกราฟได้ดังรูปที่ 3.6(ข)



(ก) (ข)

รูปที่ 3.6 ตัวอย่างโครงสร้างคำและรูปแบบกราฟของคำที่มีตัวไหล

จากรูปแบบกราฟที่เป็นตัวแทนของคำภาษาล้านนาที่ได้ออกแบบไว้นั้น มีการกำหนดรายอะเอียดของกราฟดังต่อไปนี้ กำหนดให้ป้ายชื่อของจุดยอดแทนพยัญชนะ สระ หรือตัวสะกด และป้ายชื่อของเส้นเชื่อมแทนตำแหน่งของพยัญชนะ สระ หรือตัวสะกด ที่อยู่ในคำนั้น โดยเส้นเชื่อมแต่ละเส้นแสดงความสัมพันธ์ระหว่างพยัญชนะกับสระ พยัญชนะกับวรรณยุกต์ หรือสระกับวรรณยุกต์ ซึ่งป้ายชื่อของเส้นเชื่อมมีทั้งหมด 11 ชนิด โดยมีรายละเอียดดังต่อไปนี้

1. เส้นเชื่อมประเภท after หมายถึง จุดยอดที่อยู่ปลายของเส้นเชื่อมคือ พยัญชนะที่อยู่ข้างหลังสระหน้า

2. เส้นเชื่อมประเภท before หมายถึง จุดยอดที่อยู่ปลายของเส้นเชื่อมคือ พยัญชนะที่อยู่ข้างหน้าสระหลัง

3. เส้นเชื่อมประเภท lower หมายถึง จุดยอดที่อยู่ปลายของเส้นเชื่อมคือ พยัญชนะที่อยู่ข้างล่างสระบน

4. เส้นเชื่อมประเภท upper หมายถึง จุดยอดที่อยู่ปลายของเส้นเชื่อมคือ พยัญชนะที่อยู่ข้างบนสระล่าง

5. เส้นเชื่อมประเภท tone หมายถึง จุดยอดที่อยู่ปลายของเส้นเชื่อมคือ วรรณยุกต์

6. เส้นเชื่อมประเภท has จะใช้เพื่ออธิบายจุดยอดที่อยู่ปลายของเส้นเชื่อมที่เป็นพยัญชนะหรือสระ ที่เป็นโครงสร้างเฉพาะในคำนั้นๆ

7. เส้นเชื่อมประเภท lowerchar หมายถึง จุดยอดที่อยู่ปลายของเส้นเชื่อมคือ ตัวสะกดที่อยู่ด้านล่างพยัญชนะต้น

8. เส้นเชื่อมประเภท afterchar หมายถึง จุดยอดที่อยู่ปลายของเส้นเชื่อมคือ ตัวสะกดที่อยู่ด้านหลังพยัญชนะต้น

9. เส้นเชื่อมประเภท CFC หมายถึง จุดยอดที่อยู่ปลายของเส้นเชื่อมคือ ตัวสะกดของคำที่มีพยัญชนะซ้อน

10. เส้นเชื่อมประเภท FCIC หมายถึง จุดยอดที่อยู่ปลายของเส้นเชื่อมคือ ตัวซ้อน

11. เส้นเชื่อมประเภท FCFC หมายถึง จุดยอดที่อยู่ปลายของเส้นเชื่อมคือ ตัวสะกดชั้นสองหรือตัวสะกดชั้นสาม

กราฟที่เป็นตัวแทนของโครงสร้างคำภาษาล้านนามีทั้งหมด 77 รูปแบบ โดยประกอบไปด้วย กราฟของคำผสมสระจำนวน 15 แบบ กราฟของคำที่มีตัวสะกดวางไว้ด้านล่างพยัญชนะต้นจำนวน 7 แบบ กราฟของคำที่มีตัวสะกดวางไว้ด้านหลังพยัญชนะต้นจำนวน 3 แบบ กราฟของคำที่มีพยัญชนะซ้อนจำนวน 37 แบบ และกราฟของคำที่มีตัวไหลจำนวน 15 แบบ ซึ่งสามารถดูรายละเอียดกราฟที่เป็นตัวแทนของคำได้ที่ภาคผนวก ก

3.2 การแปลงคำภาษาล้านนาให้อยู่ในรูปแบบกราฟ

คำในภาษาล้านนาแต่ละคำจะถูกแปลงให้อยู่ในลักษณะของกราฟตามประเภทของคำโดยพิจารณาจากรูปแบบกราฟที่ได้ออกแบบไว้(ดังที่กล่าวไปแล้วในหัวข้อที่ 3.1) ขั้นตอนวิธีในการแปลงคำให้อยู่ในรูปแบบกราฟสามารถแบ่งได้เป็น 5 แบบ ตามประเภทของโครงสร้างคำดังต่อไปนี้

3.2.1 ขั้นตอนวิธีการแปลงคำผสมสระให้อยู่ในรูปแบบกราฟ

ขั้นตอนการแปลงคำผสมสระให้อยู่ในรูปแบบกราฟจะเริ่มจากการตรวจสอบข้อมูลเข้าซึ่งเป็นคำภาษาล้านนาทีละตัวอักษร ดังตัวอย่างในรูปที่ 3.7 เมื่อข้อมูลเข้าคือคำว่า “ᨠᩣᩅ” การทำงานของขั้นตอนดังกล่าวอธิบายได้ดังต่อไปนี้

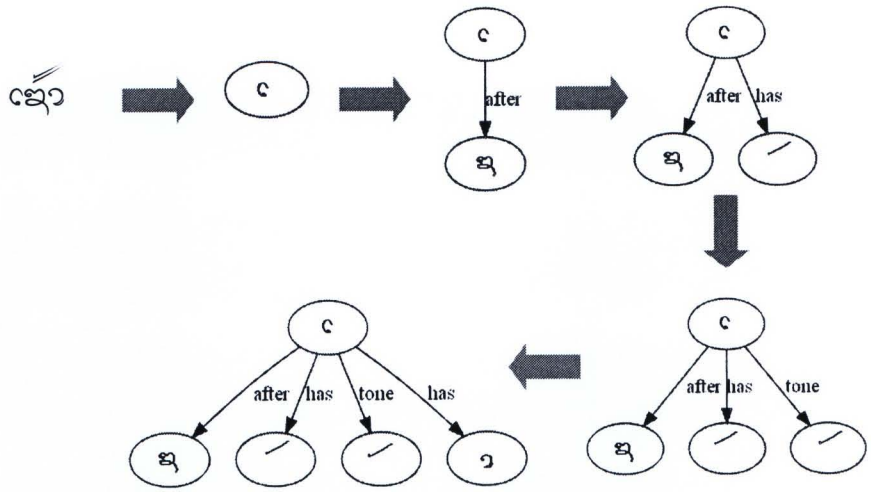
1. ตรวจสอบตัวอักษรตัวแรก คืออักษร “ᨠ” ทำการเพิ่มจุดยอดของกราฟและกำหนดป้ายชื่อของจุดยอดดังกล่าวเป็น “ᨠ”

2. ตรวจสอบตัวอักษรตัวที่สอง คืออักษร “ᩣ” ทำการเพิ่มจุดยอดของกราฟและกำหนดป้ายชื่อของจุดยอดดังกล่าวเป็น “ᩣ” จากนั้นเพิ่มเส้นเชื่อมระหว่างจุดยอด “ᨠ” กับจุดยอด “ᩣ” แล้วกำหนดป้ายชื่อของเส้นเชื่อมเป็น “after”

3. ตรวจสอบตัวอักษรตัวที่สาม คืออักษร “ ◌ᩅ ” ทำการเพิ่มจุดยอดของกราฟและกำหนดป้ายชื่อของจุดยอดดังกล่าวเป็น “◌ᩅ ” จากนั้นเพิ่มเส้นเชื่อมระหว่างจุดยอด “ᨠ” กับจุดยอด“◌ᩅ ” แล้วกำหนดป้ายชื่อของเส้นเชื่อมเป็น “has”

4. ตรวจสอบตัวอักษรตัวที่สี่ คืออักษร “◌᩵” ทำการเพิ่มจุดยอดของกราฟและกำหนดป้ายชื่อของจุดยอดดังกล่าวเป็น “◌᩵” จากนั้นเพิ่มเส้นเชื่อมระหว่างจุดยอด “ᨠ” กับจุดยอด “◌᩵” แล้วกำหนดป้ายชื่อของเส้นเชื่อมเป็น “tone”

5. ตรวจสอบตัวอักษรตัวที่ห้า คืออักษร “᩶” ทำการเพิ่มจุดยอดของกราฟและกำหนดป้ายชื่อของจุดยอดดังกล่าวเป็น “᩶” จากนั้นเพิ่มเส้นเชื่อมระหว่างจุดยอด “ᨠ” กับจุดยอด “᩶” แล้วกำหนดป้ายชื่อของเส้นเชื่อมเป็น “has”



รูปที่ 3.7 ตัวอย่างการทำงานของขั้นตอนการแปลงคำผสมสระ

จากรายละเอียดการทำงานของขั้นตอนการแปลงคำผสมสระดังที่ได้อธิบายไปแล้วข้างต้น สามารถนำมาเขียนเป็นขั้นตอนวิธีการแปลงคำผสมสระได้ตามขั้นตอนวิธีที่ 3.1

ขั้นตอนวิธีที่ 3.1 การแปลงคำผสมสระให้อยู่ในรูปแบบกราฟ

NoFinalConsonant(*word*)
Input : A Lanna word *word*
Output : A Lanna word graph *g*

```
1.  g is empty
2.  for i = 1 to length[word] do
3.      if      word[i] = ForwordVowel then
4.              add vertex word[i]
5.      else if word[i] = AboveVowel then
6.              add vertex word[i]
7.              if  i > 2 then
8.                  add edge "has"
9.              else add edge "lower"
10.     else if word[i] = BelowVowel then
11.             add vertex word[i]
12.             add edge "upper"
13.     else if word[i] = BackwordVowel then
14.             add vertex word[i]
15.             if  i > 2 then
16.                 add edge "has"
17.             else add edge "before"
```

18. else if $word[i] = Tone$ then
19. add vertex $word[i]$
20. add edge "tone"
21. else if $i > 1$ then
22. add vertex $word[i]$
23. if $word[i-1] = ForwordVowel$ then
24. add edge "after"
25. else add edge "has"
26. else add vertex $word[i]$
27. return g

ขั้นตอนวิธีที่ 3.1 เป็นการแปลงคำผสมสระให้อยู่ในรูปแบบกราฟ โดยข้อมูลเข้าของขั้นตอนวิธีนี้คือ คำภาษาล้านนา (*word*) และข้อมูลออกของขั้นตอนวิธีนี้คือ กราฟของคำภาษาล้านนา (g) ซึ่งขั้นตอนวิธีนี้จะทำการตรวจสอบตัวอักษรที่อยู่ภายในคำทีละตัวอักษร โดยในบรรทัดที่ 3-17 เป็นการตรวจสอบตัวอักษรว่าเป็นตัวอักษรที่อยู่ในกลุ่มของสระหน้า สระหลัง สระบน และสระล่างหรือไม่ เพื่อกำหนดป้ายชื่อให้กับจุดยอดและเส้นเชื่อมของกราฟ ในบรรทัดที่ 18-20 เป็นการตรวจสอบตัวอักษรว่าเป็นวรรณยุกต์หรือไม่ และในบรรทัดที่ 21-25 เป็นการตรวจสอบตัวอักษรโดยพิจารณาว่าตัวอักษรดังกล่าวเป็นพยัญชนะ หรือเป็นพยัญชนะที่ทำหน้าที่เป็นสระ

จากขั้นตอนวิธีการแปลงคำผสมสระให้อยู่ในรูปแบบกราฟ โดยพิจารณาในบรรทัดที่ 2-26 จะทำซ้ำในลูป for เท่ากับจำนวนของตัวอักษรภายในคำ ดังนั้นการแปลงคำผสมสระให้อยู่ในรูปแบบกราฟมีความซับซ้อนของเวลาเป็น $O(n)$ เมื่อ n คือจำนวนของตัวอักษรภายในคำที่เป็นข้อมูลเข้า

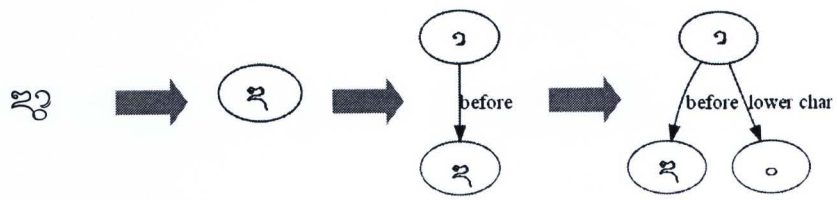
3.2.2 ขั้นตอนวิธีการแปลงคำที่มีตัวสะกดวางไว้ด้านล่างพยัญชนะต้น

ขั้นตอนการแปลงคำที่มีตัวสะกดวางไว้ด้านล่างพยัญชนะต้นให้อยู่ในรูปแบบกราฟจะเริ่มจากการตรวจสอบข้อมูลเข้าซึ่งเป็นคำภาษาล้านนาทีละตัวอักษร ดังตัวอย่างในรูปที่ 3.8 เมื่อข้อมูลเข้าคือคำว่า “ᩈᩩ” การทำงานของขั้นตอนดังกล่าวอธิบายได้ดังต่อไปนี้

1. ตรวจสอบตัวอักษรตัวแรก คืออักษร “ᩈ” ทำการเพิ่มจุดยอดของกราฟและกำหนดป้ายชื่อของจุดยอดดังกล่าวเป็น “ᩈ”

2. ตรวจสอบตัวอักษรตัวที่สอง คืออักษร “ᩩ” ทำการเพิ่มจุดยอดของกราฟและกำหนดป้ายชื่อของจุดยอดดังกล่าวเป็น “ᩩ” จากนั้นเพิ่มเส้นเชื่อมระหว่างจุดยอด “ᩩ” กับจุดยอด “ᩈ” แล้วกำหนดป้ายชื่อของเส้นเชื่อมเป็น “before”

3. ตรวจสอบตัวอักษรตัวที่สาม คืออักษร “ ◌ ” ทำการเพิ่มจุดยอดของกราฟและกำหนดป้ายชื่อของจุดยอดดังกล่าวเป็น “ ◌ ” จากนั้นเพิ่มเส้นเชื่อมระหว่างจุดยอด “ว” กับจุดยอด “ ◌ ” แล้วกำหนดป้ายชื่อของเส้นเชื่อมเป็น “lower char”



รูปที่ 3.8 ตัวอย่างการทำงานของขั้นตอนการแปลงคำที่มีตัวสะกดวางไว้ด้านล่างพยัญชนะต้น

จากรายละเอียดการทำงานของขั้นตอนการแปลงคำที่มีตัวสะกดวางไว้ด้านล่างพยัญชนะต้นดังที่ได้อธิบายไปแล้วข้างต้นสามารถนำมาเขียนเป็นขั้นตอนวิธีการแปลงคำที่มีตัวสะกดวางไว้ด้านล่างพยัญชนะต้นได้ตามขั้นตอนวิธีที่ 3.2

ขั้นตอนวิธีที่ 3.2 การแปลงคำที่มีตัวสะกดวางไว้ด้านล่างพยัญชนะต้นให้อยู่ในรูปแบบกราฟ

LowerFinalConsonant(*word*)

Input : A Lanna word *word*

Output : A Lanna word graph *g*

```
1.  g is empty
2.  for i = 1 to length[word] do
3.      if      word[i] = ForwardVowel then
4.              add vertex word[i]
5.      else if word[i] = AboveVowel then
6.              add vertex word[i]
7.              if  i > 2 then
8.                  add edge "has"
9.              else add edge "lower"
10.     else if word[i] = BackwardVowel then
11.             add vertex word[i]
12.             if  i > 2 then
13.                 add edge "has"
14.             else add edge "before"
15.     else if word[i] = Tone then
16.             add vertex word[i]
17.             add edge "tone"
```

```
18.    else if i > 1 then
19.            add vertex word[i]
20.            if   word[i − 1] = ForwardVowel  then
21.                 add edge "after"
22.            else add edge "lowerchar"
23.     else   add vertex word[i]
24. return g
```

ขั้นตอนวิธีที่ 3.2 เป็นการแปลงคำที่มีตัวสะกดวางไว้ด้านล่างพยัญชนะต้นให้อยู่ในรูปแบบกราฟ โดยข้อมูลเข้าของขั้นตอนวิธีนี้คือ คำภาษาล้านนา (*word*) และข้อมูลออกของขั้นตอนวิธีนี้คือ กราฟของคำภาษาล้านนา (*g*) โดยขั้นแรกในบรรทัดที่ 3-4 จะทำการตรวจสอบตัวอักษรว่าเป็นสระหน้าหรือไม่ ในบรรทัดที่ 5-9 ตรวจสอบว่าเป็นสระบน หรือสระที่ทำหน้าที่เป็นสระผสม ในบรรทัดที่ 10-14 เป็นการตรวจสอบตัวอักษรที่รับเข้ามาว่าเป็นสระหลัง หรือสระที่ทำหน้าที่เป็นสระผสม ในบรรทัดที่ 15-17 เป็นการตรวจสอบตัวอักษรว่าเป็นวรรณยุกต์หรือไม่ และในบรรทัดที่ 18-22 เป็นการตรวจสอบว่าตัวอักษรดังกล่าวเป็นพยัญชนะ หรือเป็นตัวสะกดล่าง

จากขั้นตอนวิธีการแปลงคำที่มีตัวสะกดวางไว้ด้านล่างพยัญชนะต้นให้อยู่ในรูปแบบกราฟ โดยพิจารณาในบรรทัดที่ 2-23 จะทำซ้ำในลูป for เท่ากับจำนวนของตัวอักษรภายในคำ ดังนั้นการแปลงคำที่มีตัวสะกดวางไว้ด้านล่างพยัญชนะต้นให้อยู่ในรูปแบบกราฟมีความซับซ้อนของเวลาเป็น $O(n)$ เมื่อ n คือจำนวนของตัวอักษรภายในคำที่เป็นข้อมูลเข้า

3.2.3 ขั้นตอนวิธีการแปลงคำที่มีตัวสะกดวางไว้ด้านหลังพยัญชนะต้น

ขั้นตอนการแปลงคำที่มีตัวสะกดวางไว้ด้านหลังพยัญชนะต้นให้อยู่ในรูปแบบกราฟจะเริ่มจากการตรวจสอบข้อมูลเข้าซึ่งเป็นคำภาษาล้านนาทีละตัวอักษร ดังตัวอย่างในรูปที่ 3.9 เมื่อข้อมูลเข้าคือคำว่า “ᨠᩥᨦ” การทำงานของขั้นตอนดังกล่าวอธิบายได้ดังต่อไปนี้

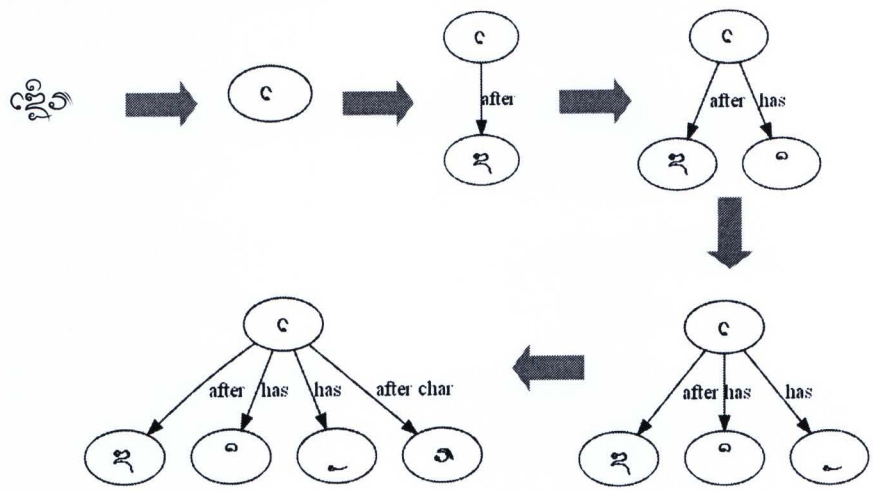
1. ตรวจสอบตัวอักษรตัวแรก คืออักษร “ᨠ” ทำการเพิ่มจุดยอดของกราฟและกำหนดป้ายชื่อของจุดยอดดังกล่าวเป็น “ᨠ”

2. ตรวจสอบตัวอักษรตัวที่สอง คืออักษร “ᨦ” ทำการเพิ่มจุดยอดของกราฟและกำหนดป้ายชื่อของจุดยอดดังกล่าวเป็น “ᨦ” จากนั้นเพิ่มเส้นเชื่อมระหว่างจุดยอด “ᨠ” กับจุดยอด “ᨦ” แล้วกำหนดป้ายชื่อของเส้นเชื่อมเป็น “after”

3. ตรวจสอบตัวอักษรตัวที่สาม คืออักษร “ ◌ิ ” ทำการเพิ่มจุดยอดของกราฟและกำหนดป้ายชื่อของจุดยอดดังกล่าวเป็น “ ◌ิ ” จากนั้นเพิ่มเส้นเชื่อมระหว่างจุดยอด “ c ” กับจุดยอด“ ◌ิ ” แล้วกำหนดป้ายชื่อของเส้นเชื่อมเป็น “has”

4. ตรวจสอบตัวอักษรตัวที่สี่ คืออักษร “ ◌ั ” ทำการเพิ่มจุดยอดของกราฟและกำหนดป้ายชื่อของจุดยอดดังกล่าวเป็น “ ◌ั ” จากนั้นเพิ่มเส้นเชื่อมระหว่างจุดยอด “ c ” กับจุดยอด “ ◌ั ” แล้วกำหนดป้ายชื่อของเส้นเชื่อมเป็น “has”

5. ตรวจสอบตัวอักษรตัวที่ห้า คืออักษร “ ◌ง ” ทำการเพิ่มจุดยอดของกราฟและกำหนดป้ายชื่อของจุดยอดดังกล่าวเป็น “ ◌ง ” จากนั้นเพิ่มเส้นเชื่อมระหว่างจุดยอด “ c ” กับจุดยอด “ ◌ง ” แล้วกำหนดป้ายชื่อของเส้นเชื่อมเป็น “after char”



รูปที่ 3.9 ตัวอย่างการทำงานของขั้นตอนการแปลงคำที่มีตัวสะกดวางไว้ด้านหลังพยัญชนะต้น

จากรายละเอียดการทำงานของขั้นตอนการแปลงคำที่มีตัวสะกดวางไว้ด้านหลังพยัญชนะต้นดังที่ได้อธิบายไปแล้วข้างต้นสามารถนำมาเขียนเป็นขั้นตอนวิธีการแปลงคำที่มีตัวสะกดวางไว้ด้านหลังพยัญชนะต้นได้ตามขั้นตอนวิธีที่ 3.3

ขั้นตอนวิธีที่ 3.3 การแปลงคำที่มีตัวสะกดวางไว้ด้านหลังพยัญชนะต้นให้อยู่ในรูปแบบกราฟ

BackwardFinalConsonant(*word*)

Input : A Lanna word *word*

Output : A Lanna word graph *g*

1. *g* is empty
2. for $i = 1$ to *length*[*word*] do
3. if *word*[*i*] = *ForwardVowel* then

```
4.          add vertex word[i]
5.    else if word[i] = BelowVowel then
6.          add vertex word[i]
7.          else add edge "upper"
8.    else if word[i] =' ◌ ' then
9.          add vertex word[i]
10.         add edge "has"
11.    else if word[i] = Tone then
12.         add vertex word[i]
13.         add edge "tone"
14.    else if i > 1 then
15.         add vertex word[i]
16.         if      word[i – 1] = ForwardVowel then
17.                 add edge "after"
18.         else if word[i] =' ◌ ' or word[i] =' ◌ ' then
19.                 add edge "has"
20.    else if word[i] =' ◌ ' then
21.         if      i > 2 then
22.                add edge "afterchar"
23.         else   add edge "before"
24.    else   add vertex word[i]
25. return g
```

ในขั้นตอนวิธีที่ 3.3 เป็นการแปลงคำที่มีตัวสะกดวางไว้ด้านหลังพยัญชนะต้นให้อยู่ในรูปแบบกราฟ ซึ่งข้อมูลเข้าของขั้นตอนวิธีนี้คือ ตัวแปร *word* ซึ่งเป็นคำภาษาล้านนาและข้อมูลออกของขั้นตอนวิธีนี้คือ กราฟของคำภาษาล้านนา (g) ซึ่งขั้นตอนวิธีนี้จะทำการตรวจสอบตัวอักษรภายในคำที่เป็นข้อมูลเข้าทีละตัวอักษร เริ่มจากในบรรทัดที่ 3-7 เป็นการตรวจสอบตัวอักษรที่รับเข้ามาว่าเป็นสระหน้า หรือสระล่างหรือไม่ ในบรรทัดที่ 8-10 ถ้าตัวอักษรที่รับเข้ามาเป็นสระอิ จะถูกกำหนดให้เป็นสระที่ทำหน้าที่เป็นสระผสม ในบรรทัดที่ 11-13 เป็นการตรวจสอบตัวอักษรว่าเป็นวรรณยุกต์หรือไม่ และในบรรทัดที่ 14-23 ตรวจสอบตัวอักษรที่รับเข้ามาว่าเป็นพยัญชนะที่ทำหน้าที่เป็นพยัญชนะต้น ทำหน้าที่เป็นสระ ทำหน้าที่เป็นสระผสม หรือทำหน้าที่เป็นตัวสะกด

จากขั้นตอนวิธีการแปลงคำที่มีตัวสะกดวางไว้ด้านหลังพยัญชนะต้นให้อยู่ในรูปแบบกราฟ โดยพิจารณาในบรรทัดที่ 2-24 จะทำซ้ำในลูป for เท่ากับจำนวนของตัวอักษรภายในคำ ดังนั้นการ

แปลงคำที่มีตัวสะกดวางไว้ด้านหลังพยัญชนะต้นให้อยู่ในรูปแบบกราฟมีความซับซ้อนของเวลาเป็น $O(n)$ เมื่อ n คือจำนวนของตัวอักษรภายในคำที่เป็นข้อมูลเข้า

3.2.4 ขั้นตอนวิธีการแปลงคำที่มีพยัญชนะซ้อนให้อยู่ในรูปแบบกราฟ

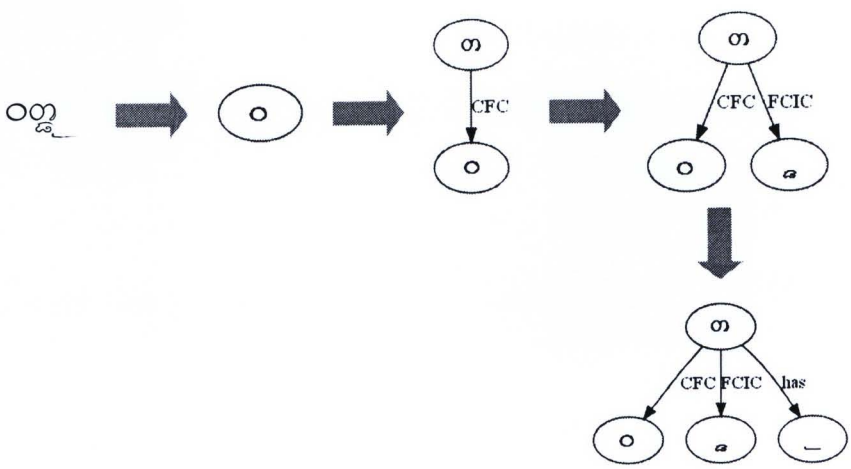
ขั้นตอนการแปลงคำที่มีพยัญชนะซ้อนให้อยู่ในรูปแบบกราฟจะเริ่มจากการตรวจสอบข้อมูลเข้าซึ่งเป็นคำภาษาล้านนาทีละตัวอักษร ดังตัวอย่างในรูปที่ 3.10 เมื่อข้อมูลเข้าคือคำว่า "ᨠᩣ" การทำงานของขั้นตอนดังกล่าวอธิบายได้ดังต่อไปนี้

1. ตรวจสอบตัวอักษรตัวแรก คืออักษร "ᨠ" ทำการเพิ่มจุดยอดของกราฟและกำหนดป้ายชื่อของจุดยอดดังกล่าวเป็น "ᨠ"

2. ตรวจสอบตัวอักษรตัวที่สอง คืออักษร "ᨣ" ทำการเพิ่มจุดยอดของกราฟและกำหนดป้ายชื่อของจุดยอดดังกล่าวเป็น "ᨣ" จากนั้นเพิ่มเส้นเชื่อมระหว่างจุดยอด "ᨣ" กับจุดยอด "ᨠ" แล้วกำหนดป้ายชื่อของเส้นเชื่อมเป็น "CFC"

3. ตรวจสอบตัวอักษรตัวที่สาม คืออักษร " ᩣ" ทำการเพิ่มจุดยอดของกราฟและกำหนดป้ายชื่อของจุดยอดดังกล่าวเป็น " ᩣ" จากนั้นเพิ่มเส้นเชื่อมระหว่างจุดยอด "ᨣ" กับจุดยอด" ᩣ" แล้วกำหนดป้ายชื่อของเส้นเชื่อมเป็น "FCIC"

4. ตรวจสอบตัวอักษรตัวที่สี่ คืออักษร " " ทำการเพิ่มจุดยอดของกราฟและกำหนดป้ายชื่อของจุดยอดดังกล่าวเป็น " " จากนั้นเพิ่มเส้นเชื่อมระหว่างจุดยอด "ᨣ" กับจุดยอด " " แล้วกำหนดป้ายชื่อของเส้นเชื่อมเป็น "has"



รูปที่ 3.10 ตัวอย่างการทำงานของขั้นตอนการแปลงคำที่มีพยัญชนะซ้อน

จากขั้นตอนการแปลงคำที่มีพยัญชนะซ้อนดังที่ได้อธิบายไปแล้วข้างต้นสามารถนำมาเขียนเป็นขั้นตอนวิธีการแปลงคำที่มีพยัญชนะซ้อนได้ตามขั้นตอนวิธีที่ 3.4

ขั้นตอนวิธีที่ 3.4 การแปลงคำที่มีพยัญชนะซ้อนให้อยู่ในรูปแบบกราฟ

DoubleFinalConsonant(*word*)
Input : A Lanna word *word*
Output : A Lanna word graph *g*

```
1. g is empty, sc = 0
2. for i = 1 to length[word] do
3.     if      word[i] = Vowel then
4.             if      word[i] = 'ว' then
5.                     add vertex word[i]
6.                     add edge "has"
7.             else if word[i] = ForwardVowel then
8.                     add vertex word[i]
9.                     add vertex word[i + 1]
10.                    add edge "has"
11.            else    add vertex word[i]
12.                    add edge "has"
13.    else if i > 1 then
14.            if      i > 2 then
15.                    if      sc = 0 then
16.                            add vertex word[i]
17.                            add edge "CFC"
18.                            sc = 1
19.                    else if i = 3 then
20.                            add vertex word[i]
21.                            add edge "FCIC"
22.                    else if word[i + 1] = TuaSon then
23.                            if      word[i] = Vowel then
24.                                    if      word[i - 2] = TuaSon then
25.                                            add vertex word[i]
26.                                            add edge "FC"
27.                                    else    add vertex word[i]
28.                                            add edge "FCIC"
29.                            else    add vertex word[i]
```

```
30.                                   add edge "FCIC"
31.                 else if     i = length[word] then
32.                                   add vertex word[i]
33.                                   add edge "C"
34.                 else              add vertex word[i]
35.                                   add edge "FCFC"
36.           else      sc = i
37.                     add vertex word[i]
38.                     add edge "CFC"
39.     else    add vertex word[i]
40. return g
```

ขั้นตอนวิธีที่ 3.4 เป็นการแปลงคำที่มีพยัญชนะซ้อนให้อยู่ในรูปแบบกราฟ ซึ่งข้อมูลเข้าของขั้นตอนวิธีนี้คือ ตัวแปร *word* ซึ่งเป็นคำภาษาล้านนาและข้อมูลออกของขั้นตอนวิธีนี้คือ กราฟของคำภาษาล้านนา (g) ซึ่งขั้นตอนวิธีนี้จะทำการตรวจสอบตัวอักษรภายในคำที่เป็นข้อมูลเข้าทีละตัวอักษร ในขั้นตอนวิธีนี้กำหนดให้ตัวแปร *sc* ทำหน้าที่เป็นตัวตรวจสอบว่าคำที่มีพยัญชนะซ้อนนั้นเป็นคำที่มีพยัญชนะซ้อนชั้นเดียว สองชั้น หรือสามชั้น ในบรรทัดที่ 3-12 เป็นการตรวจสอบตัวอักษรที่รับเข้ามาว่าเป็นสระเดี่ยวหรือสระผสม ในบรรทัดที่ 15-18 ตรวจสอบตัวอักษรว่าเป็นพยัญชนะที่ทำหน้าที่เป็นพยัญชนะต้นหรือตัวสะกด และในบรรทัดที่ 19-38 ตรวจสอบตัวอักษรที่รับเข้ามาว่าเป็นพยัญชนะที่ทำหน้าที่เป็นตัวซ้อน ตัวสะกดสองชั้น หรือตัวสะกดสามชั้น

จากขั้นตอนวิธีการแปลงคำที่มีพยัญชนะซ้อนให้อยู่ในรูปแบบกราฟ โดยพิจารณาในบรรทัดที่ 2-39 จะทำซ้ำในลูป for เท่ากับจำนวนของตัวอักษรภายในคำ ดังนั้นการแปลงคำที่มีพยัญชนะซ้อนให้อยู่ในรูปแบบกราฟมีความซับซ้อนของเวลาเป็น $O(n)$ เมื่อ n คือจำนวนของตัวอักษรภายในคำที่เป็นข้อมูลเข้า

3.2.5 ขั้นตอนวิธีการแปลงคำที่มีตัวไหลให้อยู่ในรูปแบบกราฟ

ขั้นตอนการแปลงคำที่มีตัวไหลให้อยู่ในรูปแบบกราฟจะเริ่มจากการตรวจสอบข้อมูลเข้าซึ่งเป็นคำภาษาล้านนาทีละตัวอักษร ดังตัวอย่างในรูปที่ 3.11 เมื่อข้อมูลเข้าคือคำว่า “ᨠ᩵ᩅ” การทำงานของขั้นตอนดังกล่าวอธิบายได้ดังต่อไปนี้

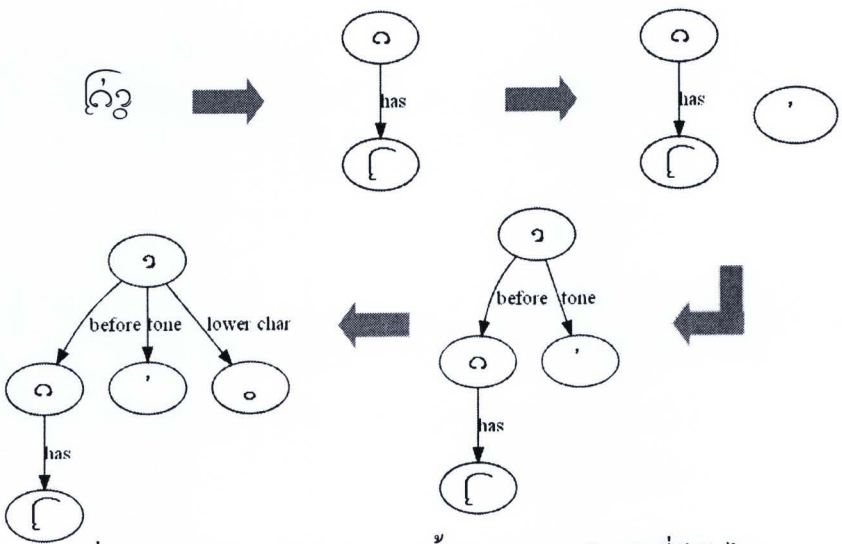
1. ตรวจสอบตัวอักษรตัวแรก คืออักษร “ᨠ” ทำการเพิ่มจุดยอดของกราฟและกำหนดป้ายชื่อของจุดยอดดังกล่าวเป็น “ᨠ”

2. ตรวจสอบตัวอักษรตัวที่สอง คืออักษร “ค” ทำการเพิ่มจุดยอดของกราฟและกำหนดป้ายชื่อของจุดยอดดังกล่าวเป็น “ค” จากนั้นเพิ่มเส้นเชื่อมระหว่างจุดยอด “ค” กับจุดยอด “ ◌ั ” แล้วกำหนดป้ายชื่อของเส้นเชื่อมเป็น “has”

3. ตรวจสอบตัวอักษรตัวที่สาม คืออักษร “ ◌่ ” ทำการเพิ่มจุดยอดของกราฟและกำหนดป้ายชื่อของจุดยอดดังกล่าวเป็น “ ◌่ ”

4. ตรวจสอบตัวอักษรตัวที่สี่ คืออักษร “ว” ทำการเพิ่มจุดยอดของกราฟและกำหนดป้ายชื่อของจุดยอดดังกล่าวเป็น “ว”จากนั้นเพิ่มเส้นเชื่อมระหว่างจุดยอด “ว” กับจุดยอด “ค” และจุดยอด “ ◌่ ” แล้วกำหนดป้ายชื่อของเส้นเชื่อมเป็น “before” และ “tone” ตามลำดับ

5. ตรวจสอบตัวอักษรตัวที่ห้า คืออักษร “ ◌ุ ” ทำการเพิ่มจุดยอดของกราฟและกำหนดป้ายชื่อของจุดยอดดังกล่าวเป็น “ ◌ุ ” จากนั้นเพิ่มเส้นเชื่อมระหว่างจุดยอด “ว” กับจุดยอด “ ◌ุ ” แล้วกำหนดป้ายชื่อของเส้นเชื่อมเป็น “lower char”



รูปที่ 3.11 ตัวอย่างการทำงานของขั้นตอนการแปลงคำที่มีตัวไหล

จากรายละเอียดการทำงานของขั้นตอนการแปลงคำตัวไหลดังที่ได้อธิบายไปแล้วข้างต้น สามารถนำมาเขียนเป็นขั้นตอนวิธีการแปลงคำที่มีตัวไหลได้ตามขั้นตอนวิธีที่ 3.5

ขั้นตอนวิธีที่ 3.5 การแปลงคำที่มีตัวไหลให้อยู่ในรูปแบบกราฟ

Cluster(*word*)
Input : A Lanna word *word*
Output : A Lanna word graph *g*

```
1.  g is empty
2.  for i = 1 to length[word] do
3.      if    word[i] = Vowel or word[i] = Tone then
4.          if    word[i] = Tone then
5.              if    i < length[word] then
6.                  if    word[i+1] = BackwardVowel or
                          word[i+1] = BelowVowel then
7.                      add vertex word[i]
8.                  else  add vertex word[i]
9.                      add edge "tone"
10.             else  add vertex word[i]
11.                 add edge "tone"
12.         else if word[i] = ForwardVowel then
13.             if    word[i+1] ='ๅ' then
14.                 add vertex word[i]
15.                 add vertex word[i+1]
16.                 add vertex word[i+2]
17.                 add edge "after"
18.                 add edge "has"
19.                 if    i > 3 then
20.                     add edge "join"
21.                 i = i + 2
22.             else  add vertex word[i]
23.                 add vertex word[i+1]
24.                 add edge "after"
25.                 if    i ≥ 3 then
26.                     add edge "join"
27.                 i = i + 1
28.         else if word[i] = AboveVowel then
29.             if    i ≥ 4 then
30.                 if    word[i-3] ='ะ' or word[i-2] ='ะ' then
31.                     add vertex word[i]
```

```
32.                         add edge "has"
33.                     else if word[i] ='ั' then
34.                         add vertex word[i]
35.                         add edge "has"
36.                     else    add vertex word[i]
37.                         add edge "lower"
38.                         if      i ≥ 4 then
39.                             write edge "join"
40.             else    add vertex word[i]
41.                     add edge "lower"
42.                     if      i > 4 then
43.                         write edge "join"
44.         else if word[i - 1] = Tone then
45.             if      word[i] = BelowVowel then
46.                     add vertex word[i]
47.                     add edge "upper"
48.                     add edge "tone"
49.                     if      i > 4 then
50.                         add edge "join"
51.             else if word[i] ='ว' then
52.                     if      word[i - 2] ='เ' then
53.                         add vertex word[i]
54.                         add edge "tone"
55.                         add edge "has"
56.                     else    add vertex word[i]
57.                         add edge "before"
58.                         add edge "tone"
59.                         if      i > 4 then
60.                             add edge "join"
61.             else    add vertex word[i]
62.                     add edge "before"
63.                     add edge "tone"
64.                     if      i > 4 then
65.                         add edge "join"
66.         else if word[i] = BackwardVowel then
67.             if      i ≥ 4 then
```

```
68.                     if      i = length[word] then
69.                             add vertex word[i]
70.                             add edge "has"
71.                     else    add vertex word[i]
72.                             add edge "before"
73.                             if      i ≥ 4 then
74.                                     add edge "join"
75.             else    add vertex word[i]
76.                     add edge "before"
77.         else    add vertex word[i]
78.                 add edge "upper"
79.                 if      i ≥ 4 then
80.                         add edge "join"
81.     else if word[i] ='ุ' then
82.             add vertex word[i]
83.             add vertex word[i + 1]
84.             add edge "has"
85.             i = i + 1
86.     else if i < length[word] and word[i] = TuaSakotLang then
87.             if      word[i - 1] = BelowVowel then
88.                     add vertex word[i]
89.                     add edge "afterchar"
90.             else    add vertex word[i]
91.                     add edge "lowerchar"
92.     else if word[i] ='ฺ ' then
93.             add vertex word[i]
94.             if      word[i - 1] = Tone then
95.                     add edge "tone"
96.                     add edge "before"
97.             else    add edge "before"
98.             if      i > 4 then
99.                     add edge "join"
100.    else if word[i] ='็ ' or word[i] ='์ ' then
101.            add vertex word[i]
102.            if      word[i - 1] = Tone then
103.                    add edge "tone"
```

104. add edge "upper"
105. else add edge "upper"
106 if $i > 4$ then
107. add edge "join"
108. else if $i <$ length[*word*] and *word*[$i+1$] = *TuaSon* then
109. add vertex *word*[i]
110. add vertex *word*[$i+1$]
111. add edge "CFC"
112. add edge "FCIC"
113. $i = i+1$
114. else if $i =$ length[*word*] then
115. add vertex *word*[i]
116. add edge "afterchar"
117. else add vertex *word*[i]
118. return g

ในขั้นตอนวิธีที่ 3.5 เป็นการแปลงคำที่มีตัวไหล ซึ่งข้อมูลเข้าของขั้นตอนวิธีนี้คือ ตัวแปร *word* ซึ่งเป็นคำภาษาล้านนาและข้อมูลออกของขั้นตอนวิธีนี้คือ กราฟของคำภาษาล้านนา (g) ซึ่งขั้นตอนวิธีนี้จะทำการตรวจสอบตัวอักษรภายในคำที่เป็นข้อมูลเข้าทีละตัวอักษร ในบรรทัดที่ 4-11 เป็นการตรวจสอบตัวอักษรว่าเป็นวรรณยุกต์หรือไม่ ในบรรทัดที่ 12-27 ทำการตรวจสอบตัวอักษรที่รับเข้ามาเป็นสระหน้าหรือสระที่ทำหน้าที่เป็นสระผสม ในบรรทัดที่ 26-43 ทำการตรวจสอบตัวอักษรที่รับเข้ามาเป็นสระบนหรือสระที่ทำหน้าที่เป็นสระผสม ในบรรทัดที่ 44-65 ถ้าตัวอักษรก่อนหน้าเป็นวรรณยุกต์ ให้ทำการตรวจสอบตัวอักษรที่รับเข้ามาว่าเป็นสระล่าง สระอา หรือสระอาที่ทำหน้าที่เป็นสระผสม ในบรรทัดที่ 66-80 เป็นการตรวจตัวอักษรที่รับเข้ามาว่าเป็นสระล่าง สระหลัง หรือสระหลังที่ทำหน้าที่เป็นสระผสม ในบรรทัดที่ 81-85 ตรวจสอบตัวอักษรที่รับเข้ามาว่าตัวถัดไปเป็นตัวระโฮงหรือไม่ ในบรรทัดที่ 86-107 ตรวจสอบพยัญชนะที่ทำหน้าที่เป็นสระ ตัวสะกดล่าง หรือตัวสะกดหลัง ในบรรทัดที่ 108-113 ตรวจสอบตัวอักษรที่รับเข้ามาว่าทำหน้าที่เป็นพยัญชนะต้น ตัวข่ม หรือตัวซ้อน ในบรรทัดที่ 114-116 ตรวจสอบตัวอักษรที่รับเข้ามาว่าเป็นตัวสะกดหลังหรือไม่

จากขั้นตอนวิธีการแปลงคำที่มีตัวไหลให้อยู่ในรูปแบบกราฟ โดยพิจารณาในบรรทัดที่ 2-117 จะทำซ้ำในลูป for เท่ากับจำนวนของตัวอักษรภายในคำ ดังนั้นการแปลงคำตัวไหลให้อยู่ในรูปแบบกราฟมีความซับซ้อนของเวลาเป็น $O(n)$ เมื่อ n คือจำนวนของตัวอักษรภายในคำที่เป็นข้อมูลเข้า