รายการอ้างอิง

1. ชมทิพ พรพนมชัย. การใช้คอมพิวเตอร์ตรวจรู้อักขระภาษาไทย. วิทยานิพนธ์ปริญญามหาบัณฑิต ภาควิชาวิศวกรรมคอมพิวเตอร์ จุฬาลงกรณ์มหาวิทยาลัย, 2529.

2. เดชา รัตนาชาร. การรู้จำตัวอักษรพิมพ์ภาษาไทยโดยใช้เทคนิคแบบฟัซซีโลจิก และ วิธีซินแทกติก. วิทยานิพนธ์ปริญญามหาบัณฑิต ภาควิชาวิศวกรรมไฟฟ้า จุฬาลงกรณ์มหาวิทยาลัย, 2538.

3. นิธิพัฒน์ ชัชวาลพาณิชย์. ระบบออนไลน์สำหรับการรู้จำตัวพิมพ์อักษรไทยและตัวพิมพ์อักษรอังกฤษ. วิทยานิพนธ์ปริญญามหาบัณฑิต ภาควิชาวิศวกรรมคอมพิวเตอร์ จุฬาลงกรณ์มหาวิทยาลัย, 2537.

4. บุญธีร์ เครือตราชู และ อภิรักษ์ จิรายุสกุล. การรู้จำตัวพิมพ์อักษรไทยโดยใช้ Counterpropogation Neural Network. การประชุมวิชาการทางวิศวกรรมไฟฟ้า ครั้งที่ 18, 2538.

5. พิพัฒน์ หิรัณย์วณิชชาการ. Recognition of thai characters. บทความวิชาการ 2530 สถาบันบัณฑิตพัฒนบริหารศาสตร์. คณะสถิติประยุกต์/ศูนย์การศึกษาระบบสารสนเทศ สถาบันบัณฑิตพัฒนบริหารศาสตร์, 2530.

6. มนลดา บุญสุวรรณ. ระบบออฟไลน์สำหรับการรู้จำตัวพิมพ์อักขระไทยหลายรูปแบบ. วิทยานิพนธ์ปริญญามหาบัณฑิต ภาควิชาวิศวกรรมคอมพิวเตอร์ จุฬาลงกรณ์มหาวิทยาลัย, 2535.

7. สนธยา เมรินทร์. การศึกษาการรู้จำตัวอักษรพิมพ์ภาษาไทยโดยวิธีซินแทกติก. วิทยานิพนธ์ปริญญามหาบัณฑิต ภาควิชาวิศวกรรมไฟฟ้า จุฬาลงกรณ์มหาวิทยาลัย, 2537.

8. Anzai, Y. Pattern Recognition and Machine learning. San Diego : Academic Press, 1989.

9. Firebaugh, M.W. Artificial Intelligence. Pws-kent, 1988.

10. Mirzai, A.R. Artificial Intelligence : Concepts and Applications in Engineering. Chapman and Hall, 1990.

11. Muggleton, S. Inductive Logic Programming. Inductive Logic Programming. San Diego : Academic Press, 1992. pp. 3-26.

12. Muggleton, S. and Feng, C. Efficient Inductive of Logic Programs. Inductive Logic Programming. San Diego : Academic Press, 1992. pp. 281-297.

13. Oxford University Computing Laboratory. Inductive Logic Programming. www@comlab.ox.ac.uk, 1998.

14. Quilan, J.R. Learning Logical Definitions from Relations. Machine Learning 5 (1990) : 239-266.

15. Sam Roberts. An Introduction to Progol. January 21, 1997.

ภาคผนวก ก

ตัวอักษรภาษาไทยที่ใช้ในการวิจัย

- ตัวอักษรที่ใช้ในการวิจัยมีทั้งหมด 77 ตัวอักษร ดังนี้

ก ข ฃ ค ฅ ฆ ง จ ฉ ช ซ ฌ ญ ฎ ฏ ฐ ฑ ฒ ณ ด ต ถ
ท ธ น บ ป ผ ฝ พ ฟ ภ ม ย ร ล ว ศ ษ ส ห ฬ อ ฮ
ะ า ิ ี ึ ื ุ ู เ แ โ ใ ไ ็ ่ ้ ๊ ๋
เ ไ ไ ๆ ฯ
๑ ๒ ๓ ๔ ๕ ๖ ๗ ๘ ๙ ๐

- รูปแบบของตัวอักษรที่ใช้ คือ CORDIA และ EUCROSIA

- ขนาดของตัวอักษรที่ใช้ คือ 20, 22, 24, 28, 32, 36 และ 48

ภาคผนวก ข

การกำหนดรูปแบบของสัญพจน์

การกำหนดรูปแบบของสัญพจน์ให้กับ PROGOL ในการเรียนรู้ตัวพิมพ์อักษรภาษาไทย ได้แสดงตัวอย่างบางส่วนพร้อมคำอธิบายไว้แล้วในบทที่ 3 หัวข้อ การเรียนรู้โดยใช้ระบบไอแอลพี ในส่วนนี้จะแสดงการกำหนดรูปแบบของสัญพจน์ทั้งหมดที่ใช้ในการเรียนรู้ ดังนี้

```
:- modeb(1,+level= #level)?
:- modeb(2,+zonelist= #zonelist)?
:- modeb(1,headzone(+sectionlist,#zone))?
:- modeb(1,headprim(+sectionlist,#prim))?
:- modeb(*,enptzone(+sectionlist,#zone))?
:- modeb(*,enptprim(+sectionlist,#prim))?
:- modeb(*,circenptzone(+sectionlist,#zone))?
:- modeb(*,circenptprim(+sectionlist,#prim))?
:- modeb(1,cntcircenpt(+sectionlist,#int))?
:- modeb(1,cntenpt(+sectionlist,#int))?
:- modeb(1,cntsection(+sectionlist,#int))?
:- modeb(1,cntprimN(+sectionlist,#int))?
:- modeb(1,cntline(+sectionlist,#int))?
:- modeb(1,cntcircle(+sectionlist,#int))?
:- modeb(1,begendzone(+sectionlist,#zone,#zone))?
:- modeb(*,memberzone(+sectionlist,#zone,#zone))?
:- modeb(*,havemember(+sectionlist,#prim,#zone,#zone))?
:- modeb(1,cntstzoneN(+sectionlist,#int))?
:- modeb(5,nozone(+zonelist))?
:- modeb(5,not nozone(+zonelist))?
:- modeb(5,havezone(+zonelist,#zn))?
:- modeb(1,cntzone_ge3(+zonelist))?
:- modeb(1,cntzone_le3(+zonelist))?
:- modeb(1,cntsection_l20(+sectionlist))?
:- modeb(1,cntsection_g3(+sectionlist))?
:- modeb(1,cntenpt_l4(+sectionlist))?
```

:- modeb(1,sizeless0_7(+float))?

:- modeb(1,sizemore1_2(+float))?

:- modeb(1,not sizemore1_2(+float))?

:- modeb(1,sizemore1_45(+float))?

:- modeb(1,not sizemore1_45(+float))?

:- modeb(1,topright_tail(+sectionlist))?

:- modeb(1,not topright_tail(+sectionlist))?

:- modeb(1,bottomright_tail(+sectionlist))?

:- modeb(1,topleft_tail(+sectionlist))?

:- modeb(1,upleft_tail(+sectionlist))?

:- modeb(1,right_line(+sectionlist))?

:- modeb(1,topline(+sectionlist))?

:- modeb(1,enpt_topright(+sectionlist))?

:- modeb(1,have4044(+sectionlist))?

:- modeb(1,not have4044(+sectionlist))?

:- modeb(1,have0011(+sectionlist))?

:- modeb(1,not have0011(+sectionlist))?

:- modeb(1,headprim_*XY*(+sectionlist))?

:- modeb(1,headprim_*XYZ*(+sectionlist))?

:- modeb(1,headzone_*XY*(+sectionlist))?

:- modeb(3,nozone0(+zonelist))?

:- modeb(3,nozone3(+zonelist))?

:- modeb(3,nozone4(+zonelist))?

:- modeb(2,headyuk(+zonelist))?

:- modeb(2,not headyuk(+zonelist))?

:- modeb(2,upperyuk(+zonelist))?

:- modeb(2,not upperyuk(+zonelist))?

ภาคผนวก ค

การกำหนดความรู้ส่วนหลัง

ในบทที่ 3 หัวข้อ การเรียนรู้โดยใช้ระบบไอแอลพี ได้กล่าวถึงความสำคัญของการกำหนดความรู้ส่วนหลัง (background knowledge) ให้กับ PROGOL และ แสดงตัวอย่างของความรู้ส่วนหลังบางส่วนไว้แล้ว ในส่วนภาคผนวกนี้จะแสดงรายละเอียดของความรู้ส่วนหลังทั้งหมดที่ใช้ในการเรียนรู้ตัวพิมพ์อักษรภาษาไทย มีรายละเอียดดังนี้

- primitive(1,0). primitive(2,0). ... primitive(1663,12). primitive(1664,12).

อนุประโยค primitive(A,B) ใช้ในการหาค่าแสดงลักษณะและทิศทางของหน่วยสร้างพื้นฐาน B (1-12) จากตัวเลขที่ใช้แทนข้อมูลส่วนย่อยของตัวอักษร A (1-1664) โดยที่

A คือ ตัวแปรชนิด sectionid

B คือ ตัวแปรชนิด prim

- endpoint(1,-1). endpoint(2,-1). ... endpoint(1663,0). endpoint(1664,0).

อนุประโยค endpoint(A,B) ใช้ในการหาค่าแสดงว่าเป็นส่วนปลาย B (-1,0) จากตัวเลขที่ใช้แทนข้อมูลส่วนย่อยของตัวอักษร A (1-1664) โดยที่

A คือ ตัวแปรชนิด sectionid

B คือ ตัวแปรชนิด enpt

- startzone(1,0). startzone(2,0). ... startzone(1663,7). startzone(1664,7).

อนุประโยค startzone(A,B) ใช้ในการหาเขตเริ่มต้นของหน่วยสร้างพื้นฐาน B (0-7) จากตัวเลขที่ใช้แทนข้อมูลส่วนย่อยของตัวอักษร A (1-1664) โดยที่

A คือ ตัวแปรชนิด sectionid

B คือ ตัวแปรชนิด zone

- endzone(1,0). endzone(2,1). ... endzone(1663,6). endzone(1664,7).

อนุประโยค endzone(A,B) ใช้ในการหาเขตสิ้นสุดของหน่วยสร้างพื้นฐาน B (0-7) จากตัวเลขที่ใช้แทนข้อมูลส่วนย่อยของตัวอักษร A (1-1664) โดยที่

A คือ ตัวแปรชนิด sectionid

B คือ ตัวแปรชนิด zone

- iscircle(8). iscircle(9). iscircle(10). iscircle(11). iscircle(12).

อนุประโยค iscircle(A) ใช้ในการระบุว่า ค่าหน่วยสร้างพื้นฐาน A มีลักษณะเป็นหน่วยสร้างพื้นฐานวงกลม โดยที่ A คือ ตัวแปรชนิด prim

- hrank(1,11). hrank(2,11). ... hrank(1663,17). hrank(1664,17).

อนุประโยค hrank(A,B) ใช้ในการหาค่าที่แสดงความน่าจะเป็นส่วนหัวของตัวอักษร B (1-17) จากตัวเลขที่ใช้แทนข้อมูลส่วนย่อยของตัวอักษร A (1-1664) ซึ่งค่า B นั้นตัวเลขที่มีค่าน้อยกว่าแสดงว่าน่าจะเป็นส่วนหัวของตัวอักษรมากกว่า โดยที่

A คือ ตัวแปรชนิด sectionid

B คือ ตัวแปรชนิด rank

- inc(A,B) :- B is A+1.

อนุประโยค inc(A,B) ใช้ในการบวกค่าเพิ่มขึ้นทีละ 1 ซึ่งจะนำไปใช้ประโยชน์ในการนับจำนวน โดย B ได้จากการบวกค่า A กับ 1 โดยที่ A และ B คือ ตัวแปรชนิด int

- iscircenpt(A) :- primitive(A,B), endpoint(A,-1), iscircle(B).

อนุประโยค iscircenpt(A) ใช้ในการระบุว่า A (1-1664) เป็นหน่วยสร้างพื้นฐานวงกลมที่เป็นส่วนปลายของตัวอักษร โดยที่ A คือ ตัวแปรชนิด sectionid

- member(A,[A|B]).
  member(A,[B|C]) :- member(A,C).

อนุประโยค member(A,B) ใช้บอกถึงความเป็นสมาชิกในลิสต์ คือ A เป็นสมาชิกในลิสต์ B โดยที่

A คือ ตัวแปรชนิดเดียวกับสมาชิกของลิสต์ B ตอนกำหนดชนิดของข้อมูล

B คือ ตัวแปรที่มีลักษณะเป็นลิสต์

- least(X,Y,X) :- hrank(X,A), hrank(Y,B), A=<B.
  least(X,Y,Y) :- hrank(X,A), hrank(Y,B), B<A.

อนุประโยค least(A,B,C) ใช้ในการเลือกหน่วยสร้างพื้นฐานที่น่าจะเป็นส่วนหัวของตัวอักษร โดยจะได้หน่วยสร้างพื้นฐาน C ซึ่ง C จะมีค่าเท่ากับหน่วยสร้างพื้นฐาน A หรือ B ขึ้นอยู่กับว่าหน่วยสร้างพื้นฐานใดน่าจะเป็นส่วนหัวของตัวอักษรมากกว่า (หรือ มีค่าที่ได้จากอนุประโยค hrank น้อยกว่า ดังที่กล่าวมาแล้วในข้อ 2.6) โดยที่

A, B และ C คือ ตัวแปรชนิด sectionid

- head([A],A).

  head([A|B],C) :- head(B,D), least(A,D,C).

อนุประโยค head(A,B) ใช้ในการหาส่วนหัวของตัวอักษร ซึ่ง A คือ ลิสต์ที่มีสมาชิกเป็นส่วนย่อยทั้งหมดของตัวอักษร ผลที่ได้คือส่วนย่อยของตัวอักษร B ซึ่งเป็นสมาชิกในลิสต์ A และ เป็นส่วนหัวของตัวอักษรนั้น โดยที่

A คือ ตัวแปรชนิด sectionlist

B คือ ตัวแปรชนิด sectionid

- headzone(A,B) :- head(A,C), startzone(C,B).

อนุประโยค headzone(A,B) ใช้ในการหาขอบเขตเริ่มต้นของส่วนหัวของตัวอักษร (B) จากลิสต์ส่วนย่อยของตัวอักษร (A) โดยที่

A คือ ตัวแปรชนิด sectionlist

B คือ ตัวแปรชนิด zone

- headprim(A,B) :- head(A,C), primitive(C,B).

อนุประโยค headprim(A,B) ใช้ในการหาค่าแสดงลักษณะและทิศทางของหน่วยสร้างพื้นฐานที่เป็นส่วนหัวของตัวอักษร (B) จากลิสต์ส่วนย่อยของตัวอักษร (A) โดยที่

A คือ ตัวแปรชนิด sectionlist

B คือ ตัวแปรชนิด prim

- enptzone(A,B) :- member(C,A), endpoint(C,-1), startzone(C,B).

อนุประโยค enptzone(A,B) ใช้บอกว่ามีสมาชิกในลิสต์ A ที่เป็นส่วนปลายของตัวอักษรและมีเขตเริ่มต้นในเขต B โดยที่

A คือ ตัวแปรชนิด sectionlist

B คือ ตัวแปรชนิด zone

- enptprim(A,B) :- member(C,A), endpoint(C,-1), primitive(C,B).

อนุประโยค enptprim(A,B) ใช้บอกว่ามีสมาชิกในลิสต์ A ที่เป็นส่วนปลายของตัวอักษรและมีค่าแสดงลักษณะและทิศทางเท่ากับ B โดยที่

A คือ ตัวแปรชนิด sectionlist

B คือ ตัวแปรชนิด prim

- circenptzone(A,B) :- member(C,A), iscircenpt(C), startzone(C,B).

อนุประโยค circenptzone(A,B) ใช้บอกว่ามีสมาชิกในลิสต์ A ที่เป็นหน่วยสร้างพื้นฐานวงกลม เป็นส่วนปลายของตัวอักษร และมีเขตเริ่มต้นในเขต B โดยที่

A คือ ตัวแปรชนิด sectionlist

B คือ ตัวแปรชนิด zone

- circenptprim(A,B) :- member(C,A), iscircenpt(C), primitive(C,B).

อนุประโยค circenptprim(A,B) ใช้บอกว่ามีสมาชิกในลิสต์ A ที่เป็นหน่วยสร้างพื้นฐานวงกลม เป็นส่วนปลายของตัวอักษร และมีค่าแสดงลักษณะและทิศทางเท่ากับ B โดยที่

A คือ ตัวแปรชนิด sectionlist

B คือ ตัวแปรชนิด prim

- cntcircenpt([],0).
  cntcircenpt([A|B],C) :- iscircenpt(A), cntcircenpt(B,D), inc(D,C).
  cntcircenpt([A|B],C) :- not iscircenpt(A), cntcircenpt(B,C).

อนุประโยค cntcircenpt(A,B) ใช้ในการนับจำนวน (B) ของหน่วยสร้างพื้นฐานวงกลมที่เป็นส่วนปลายของตัวอักษร ในลิสต์ A โดยที่

A คือ ตัวแปรชนิด sectionlist

B คือ ตัวแปรชนิด int

- cntenpt([],0).
  cntenpt([A|B],C) :- endpoint(A,-1), cntenpt(B,D), inc(D,C).
  cntenpt([A|B],C) :- endpoint(A,0), cntenpt(B,C).

อนุประโยค cntenpt(A,B) ใช้ในการนับจำนวน (B) ของหน่วยสร้างพื้นฐานที่เป็นส่วนปลายของตัวอักษร ในลิสต์ A โดยที่

A คือ ตัวแปรชนิด sectionlist

B คือ ตัวแปรชนิด int

- cntsection([],0).
  cntsection([A],1).
  cntsection([A|B],C) :- cntsection(B,D), inc(D,C).

อนุประโยค cntsection(A,B) ใช้ในการนับจำนวน (B) สมาชิกทั้งหมด ในลิสต์ A โดยที่

A คือ ตัวแปรที่มีลักษณะเป็นลิสต์

B คือ ตัวแปรชนิด int

- cntprim*N*([],0).

  cntprim*N*([A|B],C) :- primitive(A,*N*), cntprim*N*(B,D), inc(D,C).

  cntprim*N*([A|B],C) :- not primitive(A,*N*), cntprim*N*(B,C).

อนุประโยค cntprim*N*(A,B) ใช้ในการนับจำนวน (B) ของหน่วยสร้างพื้นฐานในลิสต์ A ที่มีค่าแสดงลักษณะและทิศทางเท่ากับ *N* โดยที่

A คือ ตัวแปรชนิด sectionlist

B คือ ตัวแปรชนิด int

N คือ ค่าแสดงลักษณะและทิศทางของหน่วยสร้างพื้นฐาน ดังนี้

ค่า 0-7 แทนเวกเตอร์เส้นตรง ที่ทำมุมครอบคลุมพื้นที่ 45 องศา

ค่า 8 แทนเวกเตอร์วงกลม ที่ไม่มีจุดเชื่อมต่อ

ค่า 9-12 แทนเวกเตอร์วงกลม ที่มีจุดเชื่อมต่อครอบคลุมพื้นที่ 90 องศา

- cntline([],0).

  cntline([A|B],C) :- not iscircle(A), cntline(B,D), inc(D,C).

  cntline([A|B],C) :- iscircle(A), cntline(B,C).

อนุประโยค cntline(A,B) ใช้ในการนับจำนวน (B) หน่วยสร้างพื้นฐานเส้นตรง ในลิสต์ A โดยที่

A คือ ตัวแปรชนิด sectionlist

B คือ ตัวแปรชนิด int

- cntcircle([],0).

  cntcircle([A|B],C) :- iscircle(A), cntcircle(B,D), inc(D,C).

  cntcircle([A|B],C) :- not iscircle(A), cntcircle(B,C).

อนุประโยค cntcircle(A,B) ใช้ในการนับจำนวน (B) หน่วยสร้างพื้นฐานวงกลม ในลิสต์ A โดยที่

A คือ ตัวแปรชนิด sectionlist

B คือ ตัวแปรชนิด int

- lastmember([A],A).

  lastmember([A|B],C) :- lastmember(B,C).

อนุประโยค lastmember(A,B) หมายถึง B เป็นสมาชิกตัวสุดท้ายในลิสต์ A โดยที่

A คือ ตัวแปรที่มีลักษณะเป็นลิสต์

B คือ ตัวแปรชนิดเดียวกับสมาชิกของลิสต์ A ตอนกำหนดชนิดของข้อมูล

- begendzone([A|B],C,D) :- lastmember(B,E), startzone(E,C), endzone(A,D).

อนุประโยค begendzone(A,B,C) ใช้ในการหาเขตเริ่มต้น (B) และเขตสิ้นสุด (C) ของตัวอักษรในลิสต์ A โดยที่

A คือ ตัวแปรชนิด sectionlist

B และ C คือ ตัวแปรชนิด zone

- memberzone(A,B,C) :- member(E,A), startzone(E,B), endzone(E,C).

อนุประโยค memberzone(A,B,C) หมายถึง มีส่วนย่อยของตัวอักษรที่เป็นสมาชิกในลิสต์ A มีเขตเริ่มต้นในเขต B และเขตสิ้นสุดในเขต C โดยที่

A คือ ตัวแปรชนิด sectionlist

B และ C คือ ตัวแปรชนิด zone

- havemember(A,B,C,D) :- member(E,A), primitive(E,B), startzone(E,C), endzone(E,D).

อนุประโยค havemember(A,B,C,D) หมายถึง มีส่วนย่อยของตัวอักษรที่เป็นสมาชิกในลิสต์ A มีลักษณะและทิศทางเป็นค่า B และมีเขตเริ่มต้นในเขต C และเขตสิ้นสุดในเขต D โดยที่

A คือ ตัวแปรชนิด sectionlist

B คือ ตัวแปรชนิด prim

C และ D คือ ตัวแปรชนิด zone

- cntstzone*N*([],0).

  cntstzone*N*([A|B],C) :- startzone(A,*N*), cntstzone*N*(B,D), inc(D,C).

  cntstzone*N*([A|B],C) :- not startzone(A,*N*), cntstzone*N*(B,C).

อนุประโยค cntstzone*N*(A,B) ใช้ในการนับจำนวน (B) ของส่วนย่อยของตัวอักษรในลิสต์ A ที่มีเขตเริ่มต้นในเขต *N* โดยที่

A คือ ตัวแปรชนิด sectionlist

B คือ ตัวแปรชนิด int

- nozone([]).

อนุประโยค nozone(A) หมายถึง ลิสต์ A เป็นลิสต์ที่ไม่มีสมาชิก โดยที่

A คือ ตัวแปรชนิด zonelist

- havezone(A,B) :- member(B,A).

อนุประโยค havezone(A,B) หมายถึง B เป็นสมาชิกในลิสต์ A โดยที่

A คือ ตัวแปรชนิด zonelist

B คือ ตัวแปรชนิด zn

- cntzone_ge3(A) :- cntsection(A,B), 3=<B.

อนุประโยค cntzone_ge3(A) หมายถึง ลิสต์ A มีจำนวนสมาชิกมากกว่าหรือเท่ากับ 3 โดยที่ A คือ ตัวแปรชนิด zonelist

- cntzone_le3(A) :- cntsection(A,B), B=<3.

อนุประโยค cntzone_le3(A) หมายถึง ลิสต์ A มีจำนวนสมาชิกน้อยกว่าหรือเท่ากับ 3 โดยที่ A คือ ตัวแปรชนิด zonelist

- cntsection_l20(A) :- cntsection(A,B), B<20.

อนุประโยค cntsection_l20(A) หมายถึง ลิสต์ A มีจำนวนสมาชิกน้อยกว่า 20 โดยที่ A คือ ตัวแปรชนิด sectionlist

- cntsection_g3(A) :- cntsection(A,B), 3<B.

อนุประโยค cntsection_g3(A) หมายถึง ลิสต์ A มีจำนวนสมาชิกมากกว่า 3 โดยที่ A คือ ตัวแปรชนิด sectionlist

- cntenpt_l4(A) :- cntenpt(A,B), B<4.

อนุประโยค cntenpt_l4(A) หมายถึง ตัวอักษรในลิสต์ A มีจำนวนส่วนปลายของตัวอักษรน้อยกว่า 4 ส่วน โดยที่ A คือ ตัวแปรชนิด sectionlist

- sizeless0_7(A) :- A < 0.7.

อนุประโยค sizeless0_7(A) หมายถึง อัตราส่วนความกว้างต่อความสูงของตัวอักษร (A) มีค่าน้อยกว่า 0.7 โดยที่ A คือ ตัวแปรชนิด float

- sizemore1_2(A) :- 1.2 < A.

อนุประโยค sizemore1_2(A) หมายถึง อัตราส่วนความกว้างต่อความสูงของตัวอักษร (A) มีค่ามากกว่า 1.2 โดยที่ A คือ ตัวแปรชนิด float

- sizemore1_45(A) :- 1.45 < A.

อนุประโยค sizemore1_45(A) หมายถึง อัตราส่วนความกว้างต่อความสูงของตัวอักษร (A) มีค่ามากกว่า 1.45 โดยที่ A คือ ตัวแปรชนิด float

- topright_tail(A) :- member(B,A), endpoint(B,-1), startzone(B,1), endzone (B,1), primitive(B,0).
  topright_tail(A) :- member(B,A), endpoint(B,-1), startzone(B,1), endzone (B,1), primitive(B,1).

อนุประโยค topright_tail(A) หมายถึง ตัวอักษรในลิสต์ A มีส่วนย่อยที่เป็นส่วนปลายของตัวอักษร มีลักษณะเป็นเส้นตรงทำมุม 0-90 องศา มีเขตเริ่มต้นและเขตสิ้นสุดอยู่ในเขต 1 หรืออาจกล่าวได้ว่า เป็นส่วนปลายของตัวอักษรอยู่ที่ส่วนบนด้านขวา เช่น ตัวอักษร ศ, ส เป็นต้น โดยที่ A คือ ตัวแปรชนิด sectionlist

- bottomright_tail(A) :- member(B,A), endpoint(B,-1), endzone(B,4), primitive(B,5).
  bottomright_tail(A) :- member(B,A), endpoint(B,-1), endzone(B,4), primitive(B,6).

อนุประโยค bottomright_tail(A) หมายถึง ตัวอักษรในลิสต์ A มีส่วนย่อยที่เป็นส่วนปลายของตัวอักษร มีลักษณะเป็นเส้นตรงทำมุม 225-315 องศา และมีเขตสิ้นสุดอยู่ในเขต 4 หรืออาจกล่าวได้ว่า เป็นส่วนปลายของตัวอักษรอยู่ที่ส่วนล่างด้านขวา เช่น ตัวอักษร ฯ, า เป็นต้น โดยที่ A คือ ตัวแปรชนิด sectionlist

- topleft_tail(A) :- member(B,A), endpoint(B,-1), endzone(B,2), primitive (B,4).
  topleft_tail(A) :- member(B,A), endpoint(B,-1), endzone(B,2), primitive (B,5).

อนุประโยค topleft_tail(A) หมายถึง ตัวอักษรในลิสต์ A มีส่วนย่อยที่เป็นส่วนปลายของตัวอักษร มีลักษณะเป็นเส้นตรงทำมุม 180-270 องศา และมีเขตสิ้นสุดอยู่ในเขต 2 หรืออาจกล่าวได้ว่า เป็นส่วนปลายของตัวอักษรอยู่ที่ส่วนบนด้านซ้าย เช่น ตัวอักษร ถ, ว เป็นต้น โดยที่ A คือ ตัวแปรชนิด sectionlist

- upleft_tail(A) :- member(B,A), endpoint(B,-1), startzone(B,6), endzone(B,6), primitive(B,0).
  upleft_tail(A) :- member(B,A), endpoint(B,-1), startzone(B,6), endzone(B,6), primitive(B,2).

อนุประโยค upleft_tail(A) หมายถึง ตัวอักษรในลิสต์ A มีส่วนย่อยที่เป็นส่วนปลายของตัวอักษร มีลักษณะเป็นเส้นตรงทำมุม 0-45 หรือ 90-135 องศา และ มีเขตเริ่มต้นและเขตสิ้นสุดอยู่ในเขต 6 หรืออาจกล่าวได้ว่าเป็นส่วนปลายของตัวอักษรอยู่ในเขตระดับบน เช่น ตัวอักษร ไ เป็นต้น โดยที่ A คือ ตัวแปรชนิด sectionlist

- right_line(A) :- member(B,A), endpoint(B,-1), endzone(B,1), primitive(B,1).
  right_line(A) :- member(B,A), endpoint(B,-1), endzone(B,1), primitive(B,2).

อนุประโยค right_line(A) หมายถึง ตัวอักษรในลิสต์ A มีส่วนย่อยที่เป็นส่วนปลายของตัวอักษร มีลักษณะเป็นเส้นตรงทำมุม 45-135 องศา และมีเขตสิ้นสุดอยู่ในเขต 1 หรืออาจกล่าวได้ว่า เป็นส่วนปลายของตัวอักษรที่เป็นเส้นตรงชี้ขึ้นไปยังส่วนบนด้านขวา เช่น ตัวอักษร ข, บ เป็นต้น โดยที่ A คือ ตัวแปรชนิด sectionlist

- topline(A) :- member(B,A), primitive(B,0), endpoint(B,0), startzone(B,1), endzone(B,1).
  topline(A) :- member(B,A), primitive(B,0), endpoint(B,0), startzone(B,2), endzone(B,1).

อนุประโยค top_line(A) หมายถึง ตัวอักษรในลิสต์ A มีส่วนย่อยที่ไม่ใช่ส่วนปลายของตัวอักษร มีลักษณะเป็นเส้นตรงทำมุม 0-45 องศา มีเขตเริ่มต้นในเขต 1 หรือเขต 2 และมีเขตสิ้นสุดอยู่ในเขต 1 โดยที่ A คือ ตัวแปรชนิด sectionlist

- enpt_topright(A) :- member(B,A), endpoint(B,-1), startzone(B,1), endzone (B,1).
  enpt_topright(A) :- member(B,A), endpoint(B,-1), startzone(B,2), endzone (B,1).

อนุประโยค enpt_topright(A) หมายถึง ตัวอักษรในลิสต์ A มีส่วนย่อยที่เป็นส่วนปลายของตัวอักษร มีเขตเริ่มต้นในเขต 1 หรือเขต 2 และมีเขตสิ้นสุดอยู่ในเขต 1 เช่น ตัวอักษร ร, ช เป็นต้น โดยที่ A คือ ตัวแปรชนิด sectionlist

- have4044(A) :- member(B,A), primitive(B,4), endpoint(B,0), startzone(B,4), endzone(B,4).

อนุประโยค have4044(A) หมายถึง ตัวอักษรในลิสต์ A มีส่วนย่อยที่ไม่ใช่ส่วนปลายของตัวอักษร มีลักษณะเป็นเส้นตรงทำมุม 135-180 องศา และ มีเขตเริ่มต้นและเขตสิ้นสุดอยู่ในเขต 4 โดยที่ A คือ ตัวแปรชนิด sectionlist

- have0011(A) :- member(B,A), primitive(B,0), endpoint(B,0), startzone(B,1), endzone(B,1).

อนุประโยค have0011(A) หมายถึง ตัวอักษรในลิสต์ A มีส่วนย่อยที่ไม่ใช่ส่วนปลายของตัวอักษร มีลักษณะเป็นเส้นตรงทำมุม 0-45 องศา และ มีเขตเริ่มต้นและเขตสิ้นสุดอยู่ในเขต 1 โดยที่ A คือ ตัวแปรชนิด sectionlist

- headprim_*XY*(A) :- headprim(A,*X*).
  headprim_*XY*(A) :- headprim(A,*Y*).

อนุประโยค headprim_*XY*(A) หมายถึง ตัวอักษรในลิสต์ A มีส่วนหัวของตัวอักษร ที่มีลักษณะและทิศทางของตัวอักษรเป็นค่า *X* หรือ *Y* โดยที่ A คือ ตัวแปรชนิด sectionlist

- headprim_*XYZ*(A) :- headprim(A,*X*).
  headprim_*XYZ*(A) :- headprim(A,*Y*).
  headprim_*XYZ*(A) :- headprim(A,*Z*).

อนุประโยค headprim_*XYZ*(A) หมายถึง ตัวอักษรในลิสต์ A มีส่วนหัวของตัวอักษร ที่มีลักษณะและทิศทางของตัวอักษรเป็นค่า *X* หรือ *Y* หรือ *Z* โดยที่ A คือ ตัวแปรชนิด sectionlist

- headzone_*XY*(A) :- headzone(A,*X*).
  headzone_*XY*(A) :- headzone(A,*Y*).

อนุประโยค headzone_*XY*(A) หมายถึง ตัวอักษรในลิสต์ A มีส่วนหัวของตัวอักษร ที่มีเขตเริ่มต้นอยู่ในเขต *X* หรือเขต *Y* โดยที่ A คือ ตัวแปรชนิด sectionlist

- nozone*N*(A) :- not havezone(A,z*N*).

อนุประโยค nozone*N*(A) หมายถึง ลิสต์ A ไม่มี z*N* เป็นสมาชิก โดยที่ A คือ ตัวแปรชนิด zonelist

- headyuk(A) :- member(z2,A).

อนุประโยค headyuk(A) หมายถึง ลิสต์ A มี z2 เป็นสมาชิก ใช้ในการนิยามตัวอักษรที่มีส่วนหยักอยู่ในเขต 2 หรืออาจกล่าวได้ว่า เป็นตัวอักษรที่มีหัวหยัก เช่น ตัวอักษร ข, ช เป็นต้น โดยที่ A คือ ตัวแปรชนิด zonelist

- upperyuk(A) :- member(z1,A).

upperyuk(A) :- member(z2,A).

อนุประโยค upperyuk(A) หมายถึง ลิสต์ A มี z1 หรือ z2 เป็นสมาชิก ใช้ในการนิยามตัวอักษรที่มีส่วนหยักอยู่ในเขต 1 หรือเขต 2 อาจกล่าวได้ว่า เป็นตัวอักษรที่มีส่วนบนของตัวอักษรเป็นส่วนหยัก เช่น ตัวอักษร ค, ต เป็นต้น โดยที่ A คือ ตัวแปรชนิด zonelist

ภาคผนวก ง

เพิ่มเติมรายละเอียดการใช้งานในระบบ PROGOL

การใช้งานระบบ PROGOL ได้มีกล่าวไว้แล้วในบทที่ 2 ในเนื้อหาที่จะกล่าวถึงต่อไปนี้ เป็นส่วนเพิ่มเติมรายละเอียดการใช้งานอื่นๆ ซึ่งประกอบด้วย การกำหนดค่าพารามิเตอร์ การกำหนดชนิดของข้อมูล และ การติดตั้งระบบ PROGOL

<u>การกำหนดค่าพารามิเตอร์</u> (Parameter Settings)

ผู้ใช้งานสามารถกำหนดค่าพารามิเตอร์ เพื่อควบคุมการเรียนรู้ของ PROGOL ได้ พารามิเตอร์สามารถแบ่งได้เป็น 2 กลุ่ม คือ พารามิเตอร์ที่กำหนดค่าเป็นจำนวนเต็ม (integer) โดยการใช้เพรดิเคต **set(Parameter,Value)?** ส่วนอีกกลุ่มหนึ่ง จะกำหนดค่าเป็นค่า ON หรือ OFF ซึ่งสามารถเปลี่ยนแปลงการกำหนดค่า ON หรือ OFF ได้ โดยใช้เพรดิเคต **set(Parameter)?** และ **unset(Parameter)?** ถ้าผู้ใช้งานต้องการเรียกดูค่าของพารามิเตอร์ให้ใช้เพรดิเคต **settings?**

พารามิเตอร์ที่จะกล่าวถึงต่อไปนี้ เป็นพารามิเตอร์เพียงบางส่วนที่จำเป็น ซึ่งผู้ใช้งานจะต้องทำความเข้าใจ เนื่องจากการกำหนดค่าของพารามิเตอร์ให้เหมาะสม จะช่วยเพิ่มความสามารถในการเรียนรู้ และ ช่วยให้ PROGOL ใช้ทรัพยากรได้อย่างมีประสิทธิภาพ

- set(c,Length)

คือ การกำหนดจำนวนสัญพจน์ที่มากที่สุดในส่วนเนื้อความของอนุประโยคที่ PROGOL สร้าง ดังนั้น เมื่อ PROGOL พิจารณาอนุประโยคต่างๆ จะไม่สนใจอนุประโยคที่มีจำนวนสัญพจน์มากกว่า c โดยปกติมีค่าเท่ากับ 4

- set(h,Depth)

คือ การกำหนดขอบเขตความลึกของส่วนที่ใช้พิสูจน์ว่าอนุประโยคที่ได้ครอบคลุมตัวอย่างหรือไม่ (theorem prover) โดยปกติมีค่าเท่ากับ 30

- set(I,Ival)

คือ การกำหนดลำดับชั้นที่มากที่สุดของตัวแปรใหม่ในอนุประโยค โดยปกติมีค่าเท่ากับ 3

ตัวอย่าง แสดงลำดับชั้นของตัวแปรในอนุประโยคดังต่อไปนี้

```
eastbound(A) :- nextcar(A,B), shape(B,C).
```

ตัวแปร A อยู่ในลำดับชั้น 0, ตัวแปร B อยู่ในลำดับชั้น 1 และ ตัวแปร C อยู่ในลำดับชั้น 2

- set(nodes,MaxNodes)

PROGOL จะหยุดการค้นหากฎ หลังจากที่ทำการค้นหาไปแล้วเป็นจำนวนโหนดตามที่กำหนดในพารามิเตอร์ nodes โดยที่ยังค้นหาไม่สำเร็จ โดยปกติมีค่าเท่ากับ 10000

- set(noise,MaxNegatives)

โดยปกติแล้วเราต้องการอนุประโยคที่ครอบคลุมตัวอย่างบวก โดยที่ไม่ครอบคลุมตัวอย่างลบเลย แต่ในบางกรณีก็ยอมให้ PROGOL สร้างอนุประโยคที่ครอบคลุมตัวอย่างลบจำนวนหนึ่งได้ จำนวนตัวอย่างลบดังกล่าวกำหนดได้โดยใช้พารามิเตอร์ noise โดยปกติมีค่าเท่ากับ 0

- set(verbose,Verbosity)

คือ การกำหนดความละเอียดของการแสดงผลลัพธ์ในการเรียนรู้ของ PROGOL มีค่า 0, 1 หรือ 2 ยิ่งมีค่ามาก PROGOL ก็จะแสดงผลลัพธ์ที่ละเอียดมาก โดยปกติมีค่าเท่ากับ 2

<u>การกำหนดชนิดของข้อมูล</u>

ในการกำหนดรูปแบบของสัญพจน์ จะต้องกำหนดชนิดของข้อมูลสำหรับตัวแปรทุกตัวในแต่ละเพรดิเคต ซึ่งอาจกำหนดโดยใช้ชนิดของข้อมูลมาตรฐานในโปรล็อก (Prolog) หรือ เป็นชนิดของข้อมูลที่กำหนดขึ้นเอง

ชนิดของข้อมูลมาตรฐานในโปรล็อกที่สามารถใช้ใน PROGOL ได้ มีดังนี้

- any(X)

any/1 เป็นจริง เมื่อ X คือ ค่าคงที่ ตัวแปร หรือ ฟังก์ชั่น (function)

- float(X)

float/1 เป็นจริง ถ้า X คือ จำนวนอิงดรรชนี (floating-point number)

- int(X)

int/1 เป็นจริง ถ้า X คือ เลขจำนวนเต็ม

- nat(X)

nat/1 เป็นจริง เมื่อ X คือ ตัวเลขธรรมชาติ (natural number)

- constant(X)

constant/1 เป็นจริง เมื่อ X คือ ค่าคงที่

ตัวอย่าง การกำหนดชนิดของข้อมูลขึ้นเอง เป็นชนิดลิสต์ของเลขจำนวนเต็ม

```
list([]).
list([H|T]) :- int(H), list(T).
```

<u>การติดตั้งระบบ PROGOL</u> (เวอร์ชั่น 4.2)

PROGOL สามารถใช้ในการวิจัยทางด้านการศึกษา โดยไม่เสียค่าใช้จ่าย และสามารถใช้ในการวิจัยทางด้านธุรกิจ โดยสามารถขออนุญาติ (license) ได้จาก steve@comlab.ox.ac.uk

PROGOL อยู่ในเอฟทีพีไซต์ (ftp site) ของ Oxford University Computing Laboratory ซึ่งสามารถเข้าไปได้ โดยการพิมพ์ ดังนี้

```
$ ftp ftp.comlab.ox.ac.uk
```

ให้ระบุชื่อผู้ใช้งาน (username) เป็น anonymous และ รหัสผ่าน (password) คือ ที่อยู่ไปรษณีย์อิเล็กทรอนิกส์ (e-mail address) แล้วพิมพ์คำสั่ง ดังนี้

```
ftp> get pub/Packages/ILP/progol4.2/
```

และ

```
ftp> quit
```

หลังจากนั้น จะมีสารบบ (directory) ชื่อว่า progol4.2/ ภายใต้สารบบประกอบด้วยแฟ้มข้อมูล 4 แฟ้ม คือ expand, retract, README และ progol4-2.tar.gz จากนั้น ให้พิมพ์คำสั่ง ดังนี้

```
$ expand
```

จากคำสั่งดังกล่าว จะสร้างสารบบย่อย (subdirectory) examples/ และ source/ และ แปลโปรแกรม (compile) PROGOL ในสารบบย่อย source/

ขั้นตอนสุดท้าย คือ การรวมสารบบของ PROGOL ไว้ในแฟ้มข้อมูล .login เพื่อให้สามารถเรียกใช้งาน PROGOL ได้ เหมือนกับคำสั่ง (command)

ประวัติผู้เขียน

นางสาวอภิญญา สุพรรณวรรษา เกิดวันที่ 30 ธันวาคม พ.ศ. 2515 ที่จังหวัดกรุงเทพมหานคร สำเร็จการศึกษาปริญญาตรีวิทยาศาสตรบัณฑิต สาขาศาสตร์คอมพิวเตอร์ คณะวิทยาศาสตร์และเทคโนโลยี มหาวิทยาลัยธรรมศาสตร์ ในปีการศึกษา 2536 และ เข้าศึกษาต่อในหลักสูตรวิทยาศาสตรมหาบัณฑิต ที่จุฬาลงกรณ์มหาวิทยาลัย เมื่อ พ.ศ. 2537