ระบบให้บริการแผนที่บนเว็บ

Map Services on The Web

นายเกียรติศักดิ์ คิวขุนทด
นายนรินทร์ เรืองศรี

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต
ภาควิชาวิศวกรรมคอมพิวเตอร์
คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2545

ระบบให้บริการแผนที่บนเว็บ

Map Services on The Web

โดย
นายเกียรติศักดิ์ ดิวขุนทด
นายนรินทร์ เรืองศรี

อาจารย์ที่ปรึกษา
ดร. วิศิษฏ์ หิรัญกิตติ

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต
ภาควิชาวิศวกรรมคอมพิวเตอร์
คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2545

ปริญญานิพนธ์ปีการศึกษา 2545

ภาควิชา วิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง ระบบให้บริการแผนที่บนเว็บ

Map Services on The Web

ผู้จัดทำ

1. นายเกียรติศักดิ์ คิวขุนทด รหัสประจำตัว 43015352
2. นายนรินทร์ เรืองศรี รหัสประจำตัว 43015364

_____อาจารย์ที่ปรึกษา

(ดร. วิศิษฏ์ หิรัญกิตติ)

ระบบให้บริการแผนที่บนเว็บ

นายเกียรติศักดิ์ คิวขุนทด 43015352
นายนรินทร์ เรืองศรี 43015364
ดร. วิศิษฏ์ หิรัญกิตติ อาจารย์ที่ปรึกษา
ปีการศึกษา 2545

บทคัดย่อ

แผนที่เป็นสิ่งจำเป็นอย่างยิ่งสำหรับการเดินทาง แผนที่คอมพิวเตอร์ที่มีให้บริการบนเว็บในปัจจุบันสามารถให้บริการข้อมูล เส้นทาง สถานที่ ระบบสามารถให้บริการสอบถามถนน สถานที่สำคัญว่าอยู่ตำแหน่งใดบนแผนที่ แต่ยังไม่สามารถให้บริการสอบถามเส้นทางการเดินทางจากจุดหนึ่งไปยังอีกจุดหนึ่งที่สั้นที่สุด

ปริญญานิพนธ์นี้ศึกษาหลักการทางปัญญาประดิษฐ์เพื่อใช้สร้างฐานความรู้แผนที่เพื่อให้บริการแผนที่บนเว็บโดยใช้ภาษาโปรล็อก และเพื่อให้สามารถให้บริการความรู้แผนที่บนเว็บจึงได้เลือกใช้โปรเว็บ(ProWeb) ซึ่งเป็นภาษาโปรล็อกสำหรับใช้พัฒนาโปรแกรมประยุกต์บนเครื่องเว็บเซิร์ฟเวอร์ สำหรับส่วนฐานข้อมูลของแผนที่นั้นพัฒนาโดยอาศัยโปรดาต้า (ProData)

Map Service on the Web

Keattisak Kewkhunthod 43015352
Narin Ruangsri 43015364
Dr. Visit Hirankitti Adviser

ABSTRACT

Maps are essential elements for traveling. Current web-based map services can provide positions of roads and places. The users can ask for the location of particular roads and places on the map.

In this thesis we employ artificial intelligence for developing knowledge-based map service on the web using Prolog. We use ProWeb an extension of Prolog to develop the web application on the web server and ProData to connect the system databases.

กิตติกรรมประกาศ

ปริญญานิพนธ์ฉบับนี้คงไม่อาจสำเร็จได้ด้วยดี หากไม่ได้รับความช่วยเหลือ และร่วมมือจากหลายๆฝ่ายด้วยกัน บุคคลแรกที่จะขาดเสียไม่ได้ในความสำเร็จดังกล่าวนี้ คือ อาจารย์ วิศิษฏ์ หิรัญกิตติ อาจารย์ที่ปรึกษาปริญญานิพนธ์ที่ไม่เพียงแต่สละเวลาเท่านั้น แต่ยังสละแรงกายและแรงใจในการเอาใจใส่ดูแล แนะนำ และช่วยเหลือเสมอมาจนปริญญานิพนธ์ฉบับนี้สำเร็จลุล่วงด้วยดี อีกทั้งคณาจารย์ทุกท่านท่านที่ถ่ายทอดวิชาความรู้มาให้ ซึ่งในความกรุณาของอาจารย์ที่มีต้องขอบคุณเป็นอย่างสูง

นอกจากนี้ต้องขอบคุณเจ้าหน้าที่บริษัท Logic Programming Associates ที่ช่วยให้คำปรึกษาเกี่ยวกับโปรแกรม WIN-PROLOG และต้องขอบพระคุณบุคคลสำคัญที่สุดที่ทำให้วันนี้ ก็คือ บิดา มารดา อันเป็นที่เคารพรักยิ่งซึ่งเลี้ยงดูมาเป็นอย่างดี พร้อมทั้งโอกาสในการศึกษาอย่างเต็มที่ และยังให้กำลังใจ เอาใจใส่เสมอมาในทุกๆด้าน อันที่หาที่เปรียบมิได้ ข้าพเจ้าขอระลึกในพระคุณ และขอกราบขอบพระคุณเป็นอย่างสูงมา ณ ที่นี้

นายเกียรติศักดิ์ คิวขุนทด
นายนรินทร์ เรืองศรี

สารบัญ

สารบัญภาพ

สารบัญตาราง

บทที่ 1
บทนำ

1.1 ความสำคัญและที่มา

แผนที่มีประโยชน์ในการใช้ในการอ้างอิงตำแน่งของสถานที่ การเดินทาง การติดต่อธุรกิจ หรือการท่องเที่ยว แผนที่ที่มีอยู่ไม่สามารถตอบสนองความต้องการของผู้ใช้ได้อย่างเต็มที่เช่น การค้นหาสถานที่ต่างๆต้องใช้ตารางระบุพิกัดของสถานที่ต้องการ การหาระยะทางจากจุดหนึ่งไปยังอีกจุดหนึ่งที่สั้นที่สุดไม่สามารถทำได้ จึงได้นำคอมพิวเตอร์มาแก้ปัญหาเหล่านั้น แผนที่คอมพิวเตอร์ที่มีอยู่ในปัจจุบันยังไม่สามารถให้ความรู้และข้อมูลเส้นทาง สถานที่ ได้อย่างสมบูรณ์ และหนทางหนึ่งที่จะสามารถทำให้แผนที่มีความสามารถดังกล่าวได้คือ การสร้างระบบฐานความรู้แผนที่

ภาษาโปรล็อก(Prolog) เป็นภาษาคอมพิวเตอร์ที่เหมาะสำหรับใช้ในการศึกษาหลักการทางปัญญาประดิษฐ์ เพราะมีโครงสร้างที่แน่นอน ไม่ซับซ้อน จึงใช้โปรล็อกและหลักการทางปัญญาประดิษฐ์มาใช้ในการวินิจฉัยระบบฐานข้อมูลแผนที่ ในปัจจุบันมีผู้ใช้อินเตอร์เน็ทจำนวนมากขึ้น การทำระบบให้บริการแผนที่บนเว็บจึงช่วยให้ผู้ใช้สามารถใช้งานได้จากที่ไหนก็ได้ที่สามารถเชื่อมต่ออินเตอร์เน็ทได้ แต่การนำโปรแกรมภาษาโปรล็อกให้สามารถแสดงผลบนเว็บได้จะต้องใช้โปรแกรมที่สามารถอินเตอร์พรีทเตอร์ภาษาโปรล็อกได้ และทำงานบนเว็บเซิร์ฟเวอร์เพื่อแสดงผลออกทางเว็บได้ ซึ่งเครื่องมือที่มีความสามารถนี้คือ โปรเว็บ(ProWeb) ซึ่งเป็นภาษาที่สร้างจากภาษาโปรล็อก สำหรับส่วนฐานข้อมูลของแผนที่นั้นพัฒนาโดยอาศัยโปรดาต้า (ProData)

1.2 วัตถุประสงค์ของปริญญานิพนธ์

1. ศึกษาหลักการทางปัญญาประดิษฐ์เพื่อนำมาใช้ในการสร้างแผนที่คอมพิวเตอร์เพื่อใช้งานบนเว็บเซิร์ฟเวอร์
2. ปรับปรุงแผนที่คอมพิวเตอร์ที่ให้บริการบนเว็บปัจจุบันให้มีความสามารถเพิ่มขึ้น
3. ศึกษาการใช้งานโปรเว็บเซิร์ฟเวอร์และสร้างเว็บไซต์ด้วยโปรเว็บเซิร์ฟเวอร์

1.3 ขอบเขตปริญญานิพนธ์

ปริญญานิพนธ์นี้จะสร้างเว็บให้บริการแผนที่กรุงเทพโดยใช้ภาษาโปรล็อก เครื่องมือที่ใช้ในการเขียนคือ WIN-PROLOG และ LPA PROWEB SERVER ซึ่งเป็นโปรแกรมจากบริษัท Logic Programming Associates Ltd

ระบบให้บริการแผนที่บนเว็บนี้สามารถให้บริการดังนี้

- สอบถามหาสถานที่ว่าอยู่ตำแหน่งใดหรือ หาสถานที่อยู่ในระยะที่กำหนดจากจุดที่ต้องการ
- สอบถามหาถนนว่าอยู่ที่ได้ โดยจะแสดงว่าถนนนั้นอยู่ที่ไหน
- หาเส้นทางที่สั้นที่สุดจากจุดหนึ่งไปยังอีกจุดหนึ่ง โดยจะลากเส้นแสดงเส้นทางที่วินิจฉัยได้จากการใช้หลักการค้นหาแบบฮิวริสติก (Heuristic Search) ด้วยวิธี เอ-สตาร์ (A*)

1.4 วิธีการดำเนินงาน

ปริญญานิพนธ์นี้เริ่มต้นด้วยการศึกษาทฤษฎีพื้นฐานต่างๆที่เกี่ยวข้องกับปริญญานิพนธ์ คือ

1. ศึกษาการภาษาโปรล็อกและการใช้งาน WIN-PROLOG[5]
2. ศึกษาการและการใช้งาน ProWeb[6] และ ProData[7]
3. ศึกษาระบบแผนที่[2] และโปรแกรมระบบนำร่องชาญฉลาด[1]

หลังจากนั้นจะเป็นการสร้างเว็บ โดยเนื้อหาจะอยู่ในบทที่ 5 จะอธิบายการออกแบบและสร้างส่วนติดต่อกับผู้ใช้ และสวนวินิจฉัยฐานความรู้แผนที่รวมถึงการนำแผนที่แสดงบนเว็บ ต่อมาจะเป็นการทดสอบการทำงานของระบบให้บริการแผนที่ตามหน้าที่ต่างๆที่กำหนดไว้ซึ่งจะอยู่ในบทที่ 6 บทวิเคราะห์และสรุปจะอยู่ในบทที่ 7 เป็นการสรุปการทำงานของระบบเป็นไปตามเป้าหมายหรือไม่ และแนวทางในการพัฒนาโปรแกรม

บทที่ 2
ภาษาโปรล็อก

2.1 ภาษาโปรล็อก

ภาษาคอมพิวเตอร์ที่พัฒนากันมาจากอดีตจนถึงปัจจุบันนี้ส่วนใหญ่จะเป็นภาษาที่มีการกำหนดลำดับขั้นตอนในการทำงานที่แน่นอน ซึ่งเราเรียกกันว่าเป็น Procedural Programming Language ตัวอย่างเช่น ภาษาเบสิก ภาษาซี เป็นต้น ในการใช้งานภาษาเหล่านี้ ผู้ที่จะเขียนโปรแกรมจะต้องกำหนดลำดับขั้นตอน ซึ่งเรียกว่า Procedure หรือ Algorithm สำหรับการแก้ปัญหาขึ้นเสียก่อน จากนั้นจึงเริ่มเขียนโปรแกรมให้สามารถทำงานตามลำดับขั้นตอนที่เราได้กำหนดไว้แล้วอย่างแน่นอนโดยไม่มีความยืดหยุ่นในการแปรเปลี่ยนลำดับขั้นตอนได้ ดังจะเห็นได้จากการเขียน Flow Chart ของทุกโปรแกรม จะแสดงถึงลำดับการ Execute ที่แน่นอน อย่างไรก็ตามยังมีกลุ่มของปัญหาอีกกลุ่มหนึ่ง ซึ่งไม่สามารถที่จะใช้ภาษาคอมพิวเตอร์ประเภท Procedural Programming ในการช่วยแก้ปัญหาได้ หรือกระทำได้ยากมาก และไม่มีประสิทธิภาพ กลุ่มของปัญหานี้เรียกชื่อกันว่า Artificial Intelligence (AI) หรืออาจจะอธิบายเป็นภาษาธรรมดาได้ว่าเป็นสิ่งประดิษฐ์ที่ชาญฉลาด ดังนั้นในช่วงปี 1970 จึงได้มีการพัฒนาภาษาเพื่อใช้กับ AI ขึ้น 2 ภาษาคือ ภาษา LISP และภาษาโปรล็อกซึ่งนั้นเป็นคำย่อมาจากคำว่า Programming in Logic หรือการเขียนโปรแกรมโดยวิธีทางตรรกวิทยา

ภาษาโปรล็อกมีลักษณะตรงกันข้ามกับพวก Procedural Programming ที่เด่นชัดก็คือ ความสามารถในการวินิจฉัย และสรุปความเห็นโดยการชักเหตุผลจากข้อเท็จจริง โดยอาศัยฐานความรู้(Knowledge Base) ที่มีอยู่ในโปรแกรมประกอบกับกลไกการวินิจฉัย (Inference Engine) ที่เราจะสร้างให้กับโปรแกรม โดยอาศัยการทำงานทางด้านตรรกวิทยา ดังนั้นภาษาโปรล็อกจึงเหมาะสำหรับการนำไปแก้ปัญหาที่มีโครงสร้างไม่แน่นอน ซึ่งโปรแกรมในภาษาที่ใช้อยู่ทั่วไปไม่สามารถกระทำได้ หรือสามารถทำได้แต่ไม่มีประสิทธิภาพ ในทางกลับกันภาษาโปรล็อกก็ไม่เหมาะที่จะนำไปใช้กับปัญหาที่มีการคำนวณหรือทำงานตามลำดับขั้นตอนที่แน่ชัดแล้ว อย่างไรก็ตามการพัฒนาภาษาโปรล็อกจนถึงปัจจุบันนี้ก็ยังไม่บรรลุถึงวัตถุประสงค์สุดยอดของการนำไปใช้กับ AI ซึ่งเราจะเห็นได้ว่าภาษายังมีลักษณะของ Procedural Programming อยู่บ้าง ตัวอย่างความเหมาะสมของการใช้ภาโปรล็อกใน AI คือ

- ระบบผู้เชี่ยวชาญ (Expert System)
- การโต้ตอบด้วยภาษาธรรมชาติ (Natural Language Processing)
- การสร้างเกมที่ต้องใช้การชักจูงด้วยเหตุผล
- การประยุกต์ใช้กับหุ่นยนต์อุตสาหกรรม (Robotics)

2.2 หลักการขั้นต้นของภาษาโปรล็อก

ภาษาโปรล็อก เป็นภาษาคอมพิวเตอร์ที่ใช้สำหรับการแก้ปัญหาซึ่งเกี่ยวข้องกับวัตถุ (Objects) และความสัมพันธ์ระหว่างวัตถุ (Relation) หลักการสำคัญขั้นต้นสำหรับโปรล็อกก็คือ การทำความเข้าใจกับคำว่า Facts, Rules และ Variables ส่วนใหญ่ของโปรแกรมโปรล็อกประกอบด้วยการรวบรวมฐานความรู้เกี่ยวกับเรื่องใดเรื่องหนึ่งโดยเฉพาะ ฐานความรู้นี้จะถูกเรียบเรียงอยู่ในรูปแบบของข้อเท็จจริง (Facts) และกฎ (Rules) สมมติว่าเรา

ต้องการจะบอกโปรล็อกเกี่ยวกับข้อเท็จจริงว่า เมธีชอบอรัญญา หรือ Mathee likes Arunya เราจะสามารถเขียนเป็นรูปแบบมาตรฐานในโปรล็อกดังนี้คือ

likes(mathee, arunya).

ในที่นี้ทั้ง mathee และ arunya เรียกว่าวัตถุ (Objects) ส่วน likes เรียกว่าความสัมพันธ์ (Relation) นั่นคือโปรล็อกจะบอกถึงความสัมพันธ์ระหว่างวัตถุคือเมธี และอรัญญา ว่าเมธีชอบอรัญญา สิ่งสำคัญที่เป็นข้อสังเกตคือ

1. ทั้งชื่อของวัตถุและความสัมพันธ์จะต้องขึ้นต้นด้วยตัวอักษรตัวเล็ก
2. ในการเขียนประโยค ให้เขียนความสัมพันธ์ก่อน ส่วนวัตถุให้เขียนอยู่ในวงเล็บ และแยกออกจากกันด้วยเครื่องหมายจุลภาค (,)
3. เมื่อสิ้นสุดประโยคของข้อเท็จจริงจะต้องมีเครื่องหมายมหัพภาค (.) เสมอ

ในขณะที่เราให้คำนิยามแสดงความสัมพันธ์ระหว่างวัตถุที่อยู่ภายในวงเล็บ เราจะต้องให้ความสำคัญของลำดับการเอ่ยถึงวัตถุ เช่น

likes(mathee, arunya). หมายถึง "mathee likes arunya."

likes(arunya, mathee). หมายถึง "arunya likes mathee."

ซึ่งความหมายของทั้งสองประโยคข้างบนนี้จะไม่เหมือนกัน ให้เราลองมาพิจารณาการเขียนข้อเท็จจริงในโปรล็อกอีกดังนี้

valuable(diamond). หมายถึง "diamond is valuable."

owns(mathee, diamond). หมายถึง "mathee owns diamond."

father(mathee, suvit). หมายถึง "mathee is suvit's father."

ซึ่งเราจะเห็นได้ว่า การตีความหมายของประโยคขึ้นอยู่กับการให้ความหมายของผู้เขียนโปรแกรมเอง ดังจะเห็นได้ว่า เราอาจจะตีความหมายในประโยคสุดท้ายว่า "mathee 's father is suvit." ก็ได้ ฉะนั้นในการเขียนแต่ละประโยคในโปรแกรม ผู้เขียนจะต้องให้ความหมายอย่างสมนัยกันตลอด โปรแกรม วัตถุ หรือ object ที่อยู่ภายในวงเล็บมีชื่อเรียกอีกอย่างหนึ่งว่า Argument และความสัมพันธ์ หรือ Relation ที่อยู่ข้างหน้าวงเล็บมีชื่อเรียกอีกอย่างหนึ่งว่า Functor ในแต่ละประโยคเราจะให้มีวัตถุจำนวนเท่าไดก็ได้ เช่น

watch(mathee, arunya, movie). หมายถึง "mathee and arunya watch movie."

play(mathee, thanit, tennis). หมายถึง "mathee and thanit play tennis. "

ประโยคทุกประโยคที่แสดงเป็นตัวอย่างมานี้เราเรียกว่า ข้อเท็จจริง (Facts) เมื่อเรามีข้อเท็จจริงเก็บสะสมอยู่ในโปรแกรมแล้วก็เท่ากับว่าเราได้จัดให้โปรแกรมมีความรู้ระดับหนึ่งเช่นกัน ฉะนั้นเราจึงสามารถที่จะสอบถามความรู้นี้จากโปรแกรมได้ การสอบถามหรือตั้งคำถามใน Prolog นั้น ก็คือการตั้งเป้าประสงค์ (Goal) นั่นเอง สมมติว่าเรามีข้อเท็จจริงในโปรแกรมดังกล่าวมาแล้ว เมื่อเราตั้งคำถามเราจะได้รับการตอบโต้ลักษณะดังนี้คือ

คำถาม Goal : valuable (diamond).

คำตอบ True

คำถาม Goal : likes(mathee,arunya).

คำตอบ True

คำถาม Goal : father(mathee, chaiyan).

คำตอบ False

ในที่นี้โปรล็อกจะให้คำตอบว่า "เป็นจริง" หรือ "เป็นเท็จ" โดยอาศัยข้อเท็จจริงที่มีอยู่ในโปรแกรมแล้ว การที่โปรล็อกให้คำตอบได้ถูกต้องก็เพราะว่า เมื่อเราตั้งคำถามหรือเป้าประสงค์แล้วโปรล็อกจะทำการค้นหาข้อเท็จจริงที่ตรงกับคำถามของเรา ถ้าโปรล็อกค้นพบก็จะตอบว่า True แต่ถ้าหาไม่พบก็จะตอบว่า False ให้สังเกตว่าคำถามและข้อเท็จจริงจะมีประโยคและตัวอักษรเหมือนกันทุกประการ

การตั้งคำถามเหมือนข้างบนนี้ยังไม่เป็นการคล่องตัวพอสำหรับการใช้งานโปรล็อก ในทางปฏิบัติ เราควรจะถามดังนี้

คำถาม Goal : likes(mathee, Who).

คำตอบ Who = arunya

คำถาม Goal : father(Father, Son).

คำตอบ Father = mathee

Son = suvit

ให้สังเกตว่าคำถามข้างบนนี้จะมีค่าที่อยู่ในวงเล็บ หรือ Object ซึ่งใช้อักษรตัวใหญ่นำหน้า ได้แก่ Who, Father และ Son และโปรล็อกได้ทำการค้นหาคำตอบของคำเหล่านี้ให้เราโดยการเทียบกับประโยคหรือฐานข้อมูลที่มีอยู่ในโปรแกรม คำที่ขึ้นต้นด้วยอักษรตัวใหญ่เหล่านี้ สามารถใช้แทนคำใดๆก็ได้ที่สามารถจะเข้าคู่กับคำถามได้พอดี และเราเรียกว่าตัวแปร (Variable) การมีตัวแปรที่สามารถแทนคำอื่นๆได้เช่นนี้ ทำให้โปรล็อกมีความยืดหยุ่นในการใช้งานได้ดีขึ้น

สำหรับกฎในโปรล็อกก็เป็นส่วนที่มีความสำคัญเท่าๆกับหลักการของข้อเท็จจริงที่ได้กล่าวมา เราจะกล่าวโดยสรุปในหลักการเบื้องต้นได้ว่าการเขียนโปรแกรมโปรล็อกจะประกอบด้วย

1. การบอกกล่าวถึงข้อเท็จจริงเกี่ยวกับวัตถุและความสัมพันธ์ระหว่างวัตถุ
2. การกำหนดกฎที่เกี่ยวข้องกับวัตถุและความสัมพันธ์ระหว่างวัตถุ
3. การตั้งเป้าประสงค์เกี่ยวกับวัตถุและความสัมพันธ์ระหว่างวัตถุ

2.3 ข้อเท็จจริง (Fact)

เนื่องจากโปรล็อกเป็นภาษาที่ใช้ตรรกวิทยาในการตัดสินใจ ฉะนั้นส่วนใหญ่ของโปรแกรมโปรล็อกจึงเป็นการรวบรวมฐานความรู้ (Knowledge Base) เกี่ยวกับเรื่องใดเรื่องหนึ่งโดยเฉพาะ ฐานความรู้นี้อาจจะเรียกได้ว่าเป็นฐานข้อมูล (Database) และในโปรล็อก เราจะเขียนฐานข้อมูลในรูปแบบของข้อเท็จจริง และกฎ (Facts and Rules) โปรล็อกให้เราสามารถบรรยายข้อเท็จจริงที่แสดงเป็นสัญลักษณ์ของความสัมพันธ์ได้ ตัวอย่างเช่น ถ้าเราต้องการจะบรรยายข้อเท็จจริงที่ว่า วิทยุไม่ดัง หรือ

The radio is dead.

เราจะสามารถแสดงในรูปแบบของโปรล็อกได้เป็น

is(radio, dead).

รูปแบบที่เราแสดงเป็นข้อเท็จจริงในภาษาโปรล็อกข้างบนนี้เรียกว่า อนุประโยค (clause) ให้สังเกตว่าตอนท้ายของอนุประโยคจะมีเครื่องหมายมหัพภาค . อยู่ด้วยเสมอ ในตัวอย่างนี้ is มีชื่อเรียกว่า Relation หรือความสัมพันธ์ ส่วน radio และ dead เรียกว่า วัตถุ (objects) ฉะนั้นจะเห็นได้ว่าคำว่าวัตถุไม่จำเป็นที่จะต้องเป็นวัตถุตามี่เคยเข้าใจกัน แต่จะเป็นคำนาม สรรพนาม อาการนาม กริยาหรือคำวิเศษณ์ก็ได้ วัตถุทั้งสองนี้จะขึ้นต้นด้วยตัวอักษรตัวเล็ก และมีชื่อเรียกว่า วัตถุประเภทสัญลักษณ์ (Symbol object) นอกจากต้องขึ้นต้นด้วยตัวอักษรตัวเลือกแล้ววัตถุประเภทสัญลักษณ์จะตามด้วยอักษรตัวใหญ่หรือตัวเลขก็ได้ แต่จะต้องมีจำนวนรวมกันไม่เกิน 250 ตัว และจะต้องไม่เว้นช่องว่างเอาไว้ ตัวอย่างต่อไปนี้แสดงถึงการเขียนข้อเท็จจริง และวัตถุประเภทสัญลักษณ์ที่ถูกต้อง

ภาษาอังกฤษ : The dog is white.

Manop has a computer.

Varunya likes chocolate.

ภาษาโปรล็อก :

Facts	Relation	Object
is(dog, white).	is	dog , white
has_a(manop, computer)	has_a	manop, computer
likes(varunya, chocolate).	likes	varunya, chocolate

คำบรรยายทั้งหมด ซึ่งไม่รวมถึงเครื่องหมายหัพภาพด้วย มีชื่อเรียกว่า เพรดดิเคท(Predicate) และคำที่อยู่ในวงเล็บของเพรดดิเคทเรียกว่า Argument ดังนั้นเพรดดิเคทของตัวอย่างข้างบนนี้ คือ

is(dog, white)

has_a(manop, computer)

likes(verunya, chocolate)

2.4 ตัวแปรในโปรล็อก

ตามที่ได้กล่าวมาแล้วว่า ตัวแปรในโปรล็อกจะต้องขึ้นต้นด้วยอักษรตัวใหญ่เสมอ และมีความยาวรวมกันไม่เกิน 250 อักขระ ตัวอักขระที่ต่อจากอักษรตัวแรกอาจจะเป็นตัวอักษรตัวใหญ่ ตัวเล็ก หรือตัวเลขก็ได้ ตัวอย่างเช่น

Goal : career(Profession, self_confidence)

โปรล็อกจะให้คำตอบว่า

Profession = air_hostess

Profession = engineer

2 Solutions

นั่นคือ โปรล็อกได้บอกเราว่าสำหรับผู้ที่มีความเชื่อมั่นในตนเองนั้น อาจจะมีอาชีพเป็นพนักงานต้อนรับบนเครื่องบิน หรือวิศวกรก็ได้

ให้สังเกตว่าคำว่า Profession นี้เรานำหน้าด้วยอักษรตัวใหญ่ จึงเป็นคนละคำกับ profession ที่อยู่ในโดเมน ในทำนองเดียวกันถ้าเราใช้ X แทน Profession เราก็จะได้คำตอบว่า

X = air hostess

X = engineer

2 Solutions

ในทางกลับกัน เราอาจจะถามหรือตั้งเป้าประสงค์ว่า ถ้าต้องการจะมีอาชีพเป็นแพทย์บุคคลนั้นควรจะมีคุณลักษณะอย่างไร เราจะได้คำตอบจากโปรล็อกดังนี้

Goal : career (doctor, X)

X = good_health

X = determination

X = devotion

3 Solutions

2.5 ตัวแปรที่ไม่ต้องการระบุตัวแปร

ในบางครั้งเราต้องการที่จะให้โปรล็อกไม่สนใจวัตถุบางวัตถุ (หรือบาง argument) เพื่อประโยชน์ในการควบคุมการทำงานของโปรแกรม ตัวแปรที่ไม่ระบุ จะช่วยให้เราบรรลุ ถึงวัตถุประสงค์นี้ได้ตัวแปรนี้ใช้แทนได้ด้วยเส้นใต้ (_) ดังตัวอย่าง

Goal : career (_ , self_confidence)

เราจะได้คำตอบว่า True ที่เป็นเช่นนี้ก็เพราะว่า ในวัตถุช่องแรกไม่มีตัวแปรใด ๆ (ความจริงมีตัวแปรที่ไม่ระบุ) ดังนั้นโปรล็อกจึงจับคู่เฉพาะวัตถุในช่องสุดท้ายเท่านั้น ดังนั้นการใช้ตัวแปรที่ไม่ระบุเราจะได้คำตอบแต่เพียงว่า “เป็นจริง” หรือ “เป็นเท็จ” เท่านั้น

2.6 กฎ (Rule)

ในการทำให้โปรแกรมมีประสิทธิภาพดีขึ้น เราจะต้องเพิ่มกฎ (Rules) เข้าไปในฐานข้อมูล เพื่อให้โปรล็อกสามารถใช้เป็นกลไกในการตัดสินใจโดยอาศัยข้อเท็จจริงที่มีอยู่ได้ดีขึ้น กฎก็คือประโยคที่บ่งบอกให้ทราบว่าข้อเท็จจริงข้อหนึ่งจะเป็นจริงหรือไม่จะขึ้นอยู่กับข้อเท็จจริงอื่นๆที่ตามมา ตัวอย่างเช่น เราจะบอกว่าถ้าท่านมีสุขภาพแข็งแรง มีความเชื่อมั่นในตนเอง และมีลักษณะดีแล้ว ท่านก็อาจจะเป็นพนักงาน ต้อนรับบนเครื่องบินได้

ในรูปแบบของภาษาอังกฤษเราจะเขียนได้เป็น :

IF you have good health

AND you have self_confidence.

AND you have good appearance.

THEN you probably can be an air hostess.

จากประโยคข้างบนนี้ เราจะพบว่าข้อเท็จจริงที่ว่า ท่านจะเป็นพนักงานต้อนรับบนเครื่องบินได้หรือไม่นั้นขึ้นอยู่กับข้อเท็จจริงอื่น ๆ เกี่ยวกับตัวท่านเอง ในโปรล็อกเราเรียกอนุประโยคเช่นนี้เรียกว่ากฎ ซึ่งสามารถเขียนเป็นโปรแกรมได้ดังนี้

career (air_hostess) if

qualification (good_health) and

qualification (self_confidence) and

qualification (good_appearance).

ให้สังเกตรูปแบบทั่วไปของการเขียนกฎในโปรล็อก นั่นคือ ผลสรุปจะขึ้นมาอยู่ก่อน แล้วตามด้วยคำว่า if ต่อจากนั้นจึงเป็นรายการเงื่อนไขที่ผลสรุปนี้ต้องใช้อ้างอิง โดยที่แต่ละรายการของเงื่อนไขจะถูกเชื่อมต่อกันด้วยคำว่า and และเมื่อหมดเงื่อนไขทุกรายการแล้วก็จะมีเครื่องหมายมหัพภาค . เหมือนเช่นเคย ส่วนสรุปนี้เราเรียกได้ว่าเป็นเป้าประสงค์หลัก(Goal) และเงื่อนไขแต่ละเงื่อนไขก็ถือได้ว่าเป็นเป้าประสงค์รอง(Sub Goal) เป้าประสงค์หลักมีชื่อเรียกว่า Head หรือส่วนหัว เงื่อนไขทุกเงื่อนไขรวมกันมีชื่อเรียกว่า Antecedent หรือส่วนหาง และเงื่อนไขแต่ละเงื่อนไขเรียกว่า Premise ฉะนั้นในที่นี้ส่วนหางจะประกอบด้วยสาย Premise ดังนั้นเป้าประสงค์หลักจะเป็นจริงได้ก็ต่อเมื่อเงื่อนไข หรือเป้าประสงค์รองทุกส่วนเป็นจริง แต่ถ้าเป้าประสงค์รองใดไม่เป็นจริงแล้ว เป้าประสงค์หลักก็ไม่เป็นจริงด้วย

ให้สังเกตว่าเราเขียนเป้าประสงค์รองแยกออกทีละบรรทัด ทั้งนี้เพื่อให้อ่านได้ง่าย ในทางปฏิบัติเราอาจจะเขียนไว้ในบรรทัดเดียวกันก็ได้ เนื่องจากในการเขียนโปรแกรมเราจะต้องใช้คำว่า if และ and บ่อยครั้ง ดังนั้น โปรล็อกจึงให้เราสามารถใช้สัญลักษณ์แทนได้ดังนี้ คือ

if แทนได้ด้วยสัญลักษณ์ :-

and แทนได้สัญลักษณ์ ,

or แทนได้สัญลักษณ์ ;

ฉะนั้น ในตัวอย่างข้างบน เราจึงสามารถเขียนเป็นโปรล็อกได้ดังนี้

```
carreer (air_hostess) :-
        qualification (good_health) .
        qualification (self_confidence).
        qualification (good_appearance).
```

2.7 ความจริงบางประการในการทำงานของโปรล็อก

ถึงแม้ว่าโปรล็อกจะไม่ใช่ภาษาคอมพิวเตอร์ประเภท Procedural Language แต่โปรล็อกก็ยังจะต้องทำตามขั้นตอนทั่ว ๆ ไป ดังต่อไปนี้

1. อนุประโยคที่มีเพรดดิเคทเดียวกัน จะต้องอยู่รวมกันในโปรแกรม โดยจะแทรกเพรดดิเคทอื่นเข้ามาระหว่างกลางไม่ได้ตัวอย่างเช่น เพรดดิเคท career(Person, _) จะต้องเรียงอยู่ด้วยกันเป็นต้น ทั้งนี้รวมไปถึงข้อเท็จจริงที่มีเพรดดิเคทเดียวกันด้วย
2. การทำงานของโปรล็อกเพื่อจับคู่หรือ Unification จะเริ่มจากด้านบนลงมาด้านล่าง และจากด้านซ้ายไปยังด้านขวา
3. หลังจากที่โปรล็อกพบคำตอบแรกแล้ว โปรล็อกยังคงพยายามที่จะหาคำตอบต่อไปอีกจนกว่าจะได้ค้นหาครบทุกอนุประโยคที่เกี่ยวข้องกับเป้าประสงค์

4. ถ้าเราสลับที่ของอนุประโยคในโปรแกรม โปรล็อกก็ยังคงจะให้คำตอบเช่นเดิมเสมอ แต่ประสิทธิภาพของการค้นหาคำตอบ จะขึ้นอยู่กับการเรียบเรียงตำแหน่งของอนุประโยค ดังนั้นเราจึงควรให้กฎที่มีโอกาสเกิดขึ้นน้อยที่สุดไปอยู่สุดท้ายของกฎทั้งหมด

2.8 การย้อนกลับ (Backtracking)

เป็นวิธีการค้นหาคำตอบของโปรล็อก โดยโปรล็อกจะเคลื่อนไปข้างหน้าและย้อนกลับ เพื่อพยายามหาทุกทางที่จะพิสูจน์เป้าประสงค์ให้ได้ จนกว่าจะหมดวิธีการค้นหา ลักษณะการทำงานแบบนี้ของโปรล็อกมักจะพบอยู่ในโปรแกรมเสมอ ให้ลองพิจารณาจากตัวอย่างต่อไปนี้

บุคคลในแต่ละอาชีพควรจะมีคุณลักษณะอย่างไร จึงจะเหมาะสมกับอาชีพนั้น ในที่นี้ให้พิจารณา 3 อาชีพก่อนดังนี้

Air Hostess	:	good health	(สุขภาพแข็งแรง)
		self-confidence	(มีความเชื่อมั่นในตนเอง)
		good appearance	(มีลักษณะและหน้าตาดี)
Engineer	:	good at number	(เก่งคำนวณ)
		self-confidence	(มีความเชื่อมั่นในตนเอง)
		good common sense	(มีสามัญสำนึกดี)
Doctor	:	good health	(สุขภาพแข็งแรง)
		determination	(มีความตั้งใจแน่วแน่)
		devotion	(มีความเสียสละ)

สมมติว่าบุคคลหนึ่ง มีคุณสมบัติ 2 อย่าง และเราอยากจะทราบว่าบุคคลนั้นเหมาะสมกับอาชีพอะไรโดยการถามทั้งสองเงื่อนไขพร้อมกัน ในโปรล็อกก็อนุญาตให้เราดำเนินการได้โดยตั้งเป้าประสงค์ที่มีทั้ง 2 เงื่อนไข เช่นถ้าเราต้องการให้มีคุณสมบัติว่า มีสุขภาพแข็งแรงและมีความตั้งใจแน่วแน่ เราสามารถเขียนเป็นเป้าประสงค์ได้ดังนี้

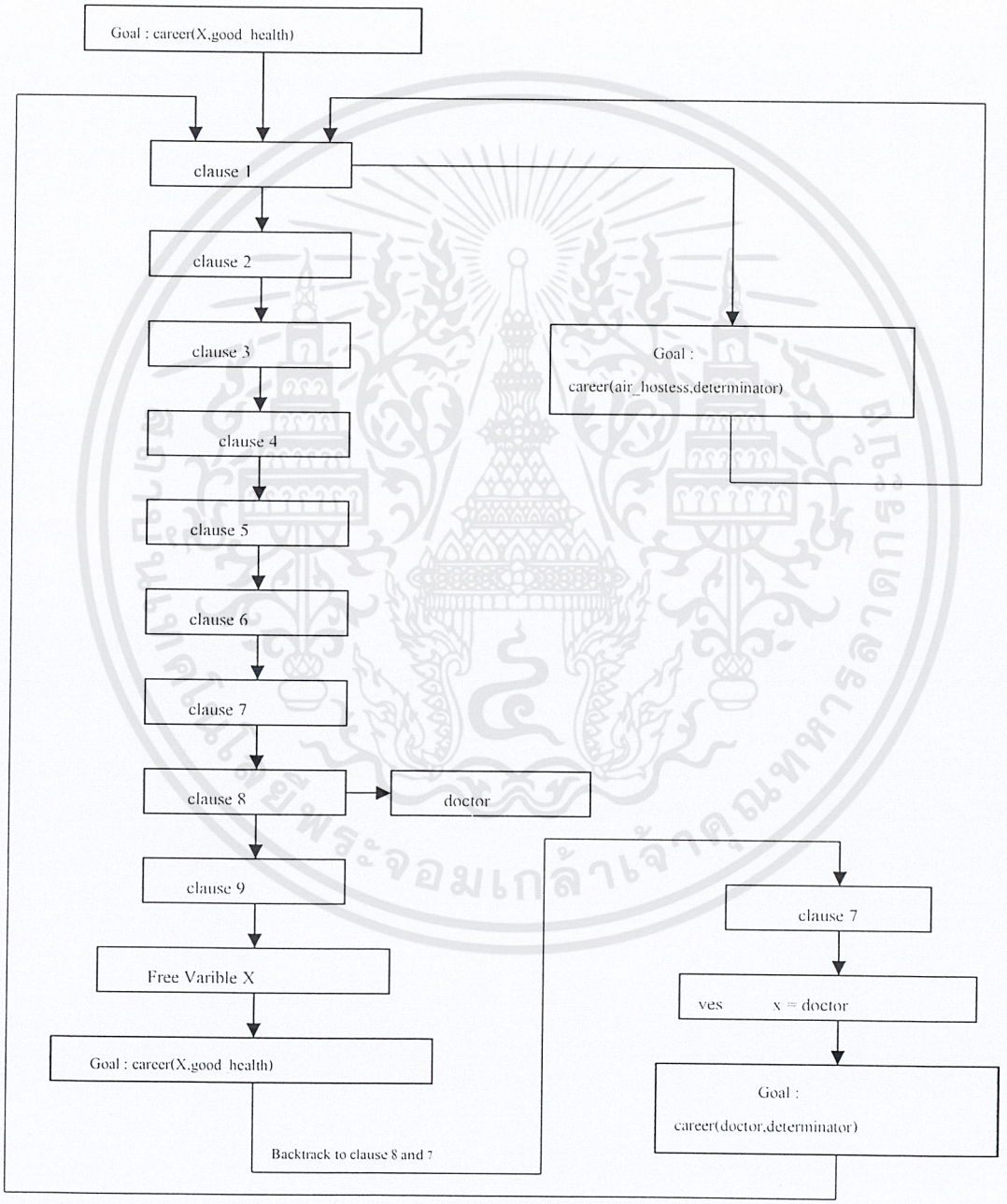
Goal : career(X, good_health)and career(X, determination)

โปรล็อกจะให้คำตอบว่า

X = doctor

การทำงานของโปรล็อกในเป้าประสงค์นี้ แสดงให้เห็นถึงคุณสมบัติหรือวิธีการหาคำตอบที่สำคัญของ โปรล็อก เรียกว่า การย้อนกลับ (Backtracking) ซึ่งสามารถอธิบายโดยพิจารณารูปที่ 2-1 ประกอบ โปรล็อกเริ่มต้นหาคำตอบโดยการพิจารณาเป้าประสงค์แรกคือ career (X, good_health) และเริ่มเทียบกับอนุประโยคที่ 1 ซึ่งปรากฏว่าสามารถจับคู่ได้พอดี ดังนั้นโปรล็อกจึงกำหนดค่าให้ตัวแปร Profession = air_hostess ก่อน จากนั้นโปรล็อกก็จะไปพิจารณาเป้าประสงค์ที่ 2 ซึ่งจะกลายเป็น career (air_hostess, determination) โปรล็อกจะพยายามพิสูจน์ความจริงนี้โดยเริ่มตั้งแต่อนุประโยคที่ 1 ใหม่ แล้วเลื่อนลงมาเรื่อย ๆจนถึงอนุประโยคที่ 9 ซึ่งเป็นอนุประโยคสุดท้าย โปรล็อกก็ยังไม่สามารถจับคู่กับอนุประโยคใดได้ ดังนั้นจึงถือว่าการค้นหาในครั้งนี้ fail หรือไม่ประสบผลสำเร็จ เมื่อเป็นเช่นนี้ โปรล็อกจะปล่อยตัวแปร Profession ให้เป็นอิสระจาก air_hostess แล้วไปพิจารณา

เป้าประสงค์แรกใหม่พร้อมกับพิจารณาย้อนกลับจากอนุประโยคที่ 9 ไป 8 และ 7 ซึ่งโปรล็อกจะพบว่าสามารถจับคู่เป้าประสงค์แรกคือ career (X, good_health) กับ career (doctor, good_health) ได้พอดี ฉะนั้นโปรล็อกจึงกำหนดให้ X = doctor จากนั้นโปรล็อกก็จะไปพิจารณาเป็นเป้าประสงค์ที่ 2 ใหม่ ซึ่งขณะนี้จะกลายเป็น career (doctor, determination) โดยย้อนกลับไปเริ่มตรวจสอบตั้งแต่อนุประโยคที่ 1 ว่าจะมีอนุประโยคใดจับคู่ได้พอดีหรือไม่ จนกระทั่งถึงอนุประโยคที่ 8 ก็พบอนุประโยคที่จับคู่ได้พอดี เมื่อโปรล็อกสามารถจับคู่หรือพิสูจน์เป้าประสงค์ทั้งสองนี้ได้ครบถ้วน ก็จะสรุปคำตอบออกมาว่า X = doctor



รูปที่ 2-1 ตัวอย่างแสดงการย้อนกลับ (Backtracking)

2.9 เพรดดิเคทเฟล (Fail)

เฟล (fail) เป็น Built- in Predicate ที่มีอยู่ในโปรล็อก เมื่อโปรล็อกพบเฟลในอนุประโยคใดก็จะมีผลให้อนุประโยคนั้นเป็นเท็จทันทีซึ่งจะทำให้โปรล็อกต้องย้อนกลับหรือBacktrackingไปยังอนุประโยคที่เหมาะสมต่อไปจะเห็นได้ว่าเฟลเป็นคำสั่งหรือเพรดดิเคทประเภทที่ใช้ควบคุมการทำงานของโปรล็อกเพื่อให้ประสบความสำเร็จโดยการบังคับหรือเราอาจจะพูดได้ว่าเป็นการใช้ความไม่สำเร็จช่วยให้เกิดความสำเร็จ ให้เราสังเกตโครงสร้างทั่วไปของโปรแกรมซึ่งใช้ fail ว่ากฎที่มีเพรดดิเคท fail อยู่จะไม่ประสบผลสำเร็จในการพิสูจน์เป้าประสงค์ทุกครั้ง และเพื่อให้โปรแกรมสามารถทำงานได้อย่างเหมาะสม เราจะต้องตามกฎนี้ด้วยอนุประโยคที่เป็นจริงเสมอ อนุประโยคเหล่านี้เรียกว่า terminating Condition นั่นคือกฎแรกจะทำงานทั้งหมดตามที่เราต้องการ ส่วนอนุประโยคที่เป็นจริงเสมอซึ่งอยู่ถัดมาจะหยุดการถอยย้อนกลับของโปรล็อก

เราสามารถสรุปการทำงานของโปรล็อกได้ว่า ในกรณีที่กฎมากกว่าหนึ่งกฎที่มีส่วนหัวเหมือนกัน และต่างก็จบลงด้วยเพรดดิเคท fail นั้น หลังจากที่โปรล็อกพยายามพิสูจน์กฎแรกจนครบทุกเพรดดิเคท หรือเงื่อนไขแล้วโปรล็อกก็จะเลื่อนลงมาที่กฎต่อไปซึ่งมีส่วนหัวเหมือนกัน และพยายามพิสูจน์กฎเช่นเดียวกันจนกระทั่งโปรล็อกเลื่อนลงมาถึงอนุประโยคหรือข้อเท็จจริง ซึ่งไม่มีเพรดดิเคท fail อยู่เป้าประสงค์จึงจะประสบผลสำเร็จ

2.10 เพรดดิเคทคัท (Cut)

คัท (Cut) เป็น built-in Predicate ที่มีความสำคัญมากที่สุดเพรดดิเคทหนึ่ง และเป็นเพรดดิเคทที่มีการใช้งานที่ค่อนข้างจะซับซ้อน จนบางครั้งเราอาจจะติดตามการทำงานของโปรล็อกได้ยาก การจะเข้าใจเพรดดิเคทคัทได้ดี ผู้อ่านจะต้องเข้าใจเกี่ยวกับการถอยย้อนกลับหรือ Backtracking ของโปรล็อกเป็นอย่างดีแล้วด้วย

วัตถุประสงค์หลักของคัทก็คือการป้องกันมิให้มีการถอยย้อนกลับหลังจากที่โปรล็อกได้พิสูจน์บางเงื่อนไขหรือเพรดดิเคทได้แล้วเพรดดิเคทคัทนี้ใช้แทนด้วยสัญลักษณ์หรือเครื่องหมาย ! เมื่อโปรล็อกเลื่อนมาจนถึงคัทแล้วโปรล็อกจะผ่านคัทไปได้เสมอ นั่นคือ เพรดดิเคทคัท จะให้ผลที่เป็นจริงเสมอ แต่เมื่อผ่านคัทไปแล้วโปรล็อกจะผ่านคัทมาใหม่ไม่ได้ และโปรล็อกจะต้องข้ามไปยังอนุประโยคอื่นที่เหมาะสมต่อไป แต่ถ้าคัทเป็นเพรดดิเคทสุดท้าย และไม่มีกฎใด ๆ ที่โปรล็อกจะต้องพยายามพิสูจน์ต่อไปอีกแล้ว โปรล็อกก็จะรับผลตามที่ได้ในขณะนั้นไม่ว่าผลนั้นจะเป็นจริงหรือเท็จ

การใช้คัทเป็นการบังคับโปรล็อกให้ยอมรับผลที่ได้ในขณะนั้นโดยไม่ต้องกลับไปตรวจสอบอนุประโยคอื่นๆอีก ดังนั้นการใช้คัทในที่ที่เหมาะสม จึงช่วยให้โปรแกรมทำงานได้ตามวัตถุประสงค์ เพรดดิเคทเฟลและคัทที่กล่าวมานี้ต่างก็มีลักษณะการบังคับการทำงานของโปรล็อก ซึ่งเข้าลักษณะของ Procedural Programming อันขัดต่อหลักการของโปรล็อกแต่ก็จำเป็นที่จะต้องมีอยู่เพื่อให้โปรแกรมสามารถทำงานได้ในบางกรณีดังเช่นตัวอย่างเหล่านี้ และยังจัดได้ว่าทั้งเพรดดิเคทเฟลและคัทต่างก็มีความสำคัญมากในโปรล็อกอีกด้วย

2.11 การตัดสินใจในการใช้คัท

วัตถุประสงค์พื้นฐานของเพรดดิเคทคัท ก็คือการช่วยให้โปรล็อกสามารถตัดทอนทิศทางการค้นหาเพื่อพิสูจน์ความจริงบางทิศทางออกไป ซึ่งอาจจะช่วยให้โปรแกรมทำงานได้เร็วขึ้น ซึ่งเราสามารถที่จะทำความเข้าใจโดยสรุปจากกฎต่อไปนี้คือ

```
run :-
        predicatel, predicate2 ,!,
        predicate3  , predicate4 ,!,
```

หลักการทำงานของโปรล็อก คือ โปรล็อกจะพยายามพิสูจน์โดยการเลื่อนย้อนกลับไปมาระหว่าง predicate1 และ predicate2 ตามความจำเป็นจนกระทั่งทั้งสองเพรดดิเคทเป็นจริง เมื่อทั้งสองเพรดดิเคทได้รับการพิสูจน์ไปแล้ว โปรล็อกก็จะมาถึงคัทเนื่องจากคัทจะเป็นจริงเสมอ ดังนั้นโปรล็อกจึงเลื่อนต่อมายัง predicate3 หลังจากที่โปรล็อกเลือกผ่านคัทมาแล้ว โปรล็อกจะไม่สามารถถอยย้อนกลับผ่านคัทไปได้อีก ดังนั้นโปรล็อกจึงต้องพิสูจน์กลับไปกลับมาระหว่าง predicate3 และ predicate4 ถ้า predicate3 หรือ predicate4 ไม่เป็นจริงกฎ run ก็จะไม่เป็นจริงด้วย แต่ก่อนที่โปรเลือกจะผ่านคัทมาค่าต่าง ๆ ของตัวแปรจะถูกกำหนดโดย Predicatel และ predicate2 แล้วฉะนั้นโปรล็อกจะให้ค่าเหล่านี้โดยอัตโนมัติ การใช้คัทนี้นอกจากจะใช้ร่วมกับเฟลดังตัวอย่างที่แสดงมาแล้วยังมักจะใช้ร่วมกับการอ้างตนเอง(Recursion)อีกด้วย ในขณะนี้เราอาจจะสรุปได้ว่า คัทอาจจะใช้สำหรับวัตถุประสงค์ใดวัตถุประสงค์หนึ่งต่อไปนี้

1. เพื่อหยุดการค้นหาคำตอบเพิ่มเติมหลังจากที่กฎใดกฎหนึ่งที่กำหนดไว้ได้รับการทดสอบแล้ว
2. เพื่อหยุดการค้นหาคำตอบเพิ่มเติมเหลังจากที่กฎใดกฎหนึ่งที่กำหนดไว้ถูกบังคับให้เป็นเท็จโดยเพรด - ดิเคทเฟล
3. สำหรับจัดทิศทางค้นหาในฐานข้อมูลบางทิศทาง เพื่อช่วยให้การทำงานของไปรล็อกเร็วขึ้น

บทที่ 3
โปรเว็บเซิร์ฟเวอร์และโปรดาต้า

3.1 โปรเว็บเซิร์ฟเวอร์

โปรเว็บเซิร์ฟเวอร์(LPA ProWeb Server : ProWeb) หรือเรียกว่าโปรเว็บ คือ HTTP เซิร์ฟเวอร์ประเภทหนึ่ง เขียนจากภาษาโปรล็อก โปรเว็บจะทำการเชื่อมต่อระหว่างเว็บเพ็จที่แสดงอยู่ที่ไคเอนท์กับโปรแกรมที่อยู่บนเซิร์ฟเวอร์ การติดต่ออันชาญฉลาด(intelligence interactive)จะเกิดขึ้นเมื่อไคเอนท์เรียกการติดต่อไปยังโปรเว็บเพื่อของขอเว็บเพ็จหน้าถัดไป โปรเว็บจะทำการส่งหน้าเพ็จไปยังไคเอนท์อย่างชาญฉลาด

โปรเว็บจะใช้ความสามารถของ WIN-PROLOG และทูลคิทให้ทำงานอยู่เบื้องหลัง โปรล็อกเป็นเทคโนโลยีที่มีประสิทธิภาพในการทำงานอย่างชาญลาด โปรเว็บใช้ความสามารถเหล่านี้ในการสร้างเอเจนต์ชาญฉลาด (intelligence agent), ระบบวินิจฉัย (diagnostic system) และอื่นอีกที่เกี่ยวกับการทำงานอย่างชาญฉลาด

ไคลเอนต์ในโปรเว็บจะหมายถึงเว็บบราวเซอร์ที่ใช้ในการติดต่อสื่อสาร และร้องขอข้อมูลจากโปรแกรมที่ทำงานอยู่บนเว็บเซิร์ฟเวอร์ โปรเว็บเป็นการเพิ่มความสามารถให้กับเว็บเซิร์ฟเวอร์ โดยทำให้เว็บเซิร์ฟเวอร์สามารถใช้งานโปรแกรมที่เขียนด้วยภาษาโปรล็อกได้ โปรเว็บอนุญาตให้หน้า HTML ที่แสดงอยู่ที่เว็บบราวเซอร์ของไคเอนท์สามารถเชื่อมโยงกับโปรแกรมที่เขียนด้วยภาษาโปรล็อกที่ทำงานอยู่บนเซิร์ฟเวอร์ ถ้าไคลเอนท์ดูหน้า HTML ที่แสดงอยู่ที่เว็บบราวเซอร์ธรรมดามันจะไม่มีการกระทำใดๆ ระหว่างไคลเอนท์กับเซิร์ฟเวอร์ โดยทั่วไปไคลเอนท์จะสามารถขอหน้า HTML หน้าถัดไปผ่านทาง URL และเซิร์ฟเวอร์จะเป็นผู้ทำหน้าที่ส่ง HTML หน้าถัดไปมาให้ถ้าใช้โปรเว็บไคลเอนท์จะสามารถสนทนากับโปรแกรมโปรเว็บที่ทำงานอยู่บนเซิร์ฟเวอร์ได้โดยตรงอย่างมีประสิทธิภาพ สามารถใช้หน้า HTML เหมือนเป็นทางผ่านของการกระทำระหว่างกันในการสนทนา

3.1.1 การทำงานย้อนกลับบนเว็บ(Backtracking on the web)

โปรเว็บจะเก็บตำแหน่งของสถานการณ์ทำงานของแต่ละไคเอนท์และใช้ข้อมูลเหล่านี้ในการทำย้อนกลับ โดยเรียกวิธีการนี้ว่า "Save State" โปรเว็บในเวอร์ชั่นนี้จะทำการประมวลผลใหม่ทุกครั้งที่มีการตอบสนองจากไคเอนท์ จะทำให้เกิดโอเวอร์เฮดมาก

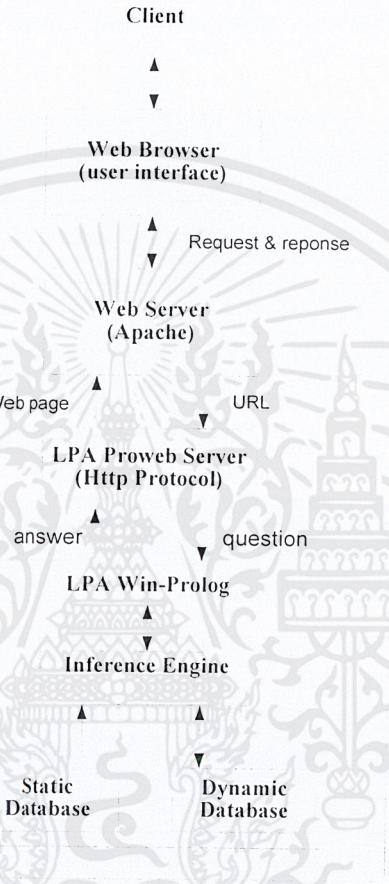
3.1.2 การให้บริการผู้ใช้จำนวนมาก(Serving multiple client)

โดยปกติเว็บจะสามารถรองรับผู้ใช้ได้หลายคนพร้อมกันที่ดูหน้า HTML เดียวกัน หมายความว่าทุกไคเอนท์สามารถคำร้องขอที่ต่างกันไปยังโปรกรมในเวลาเดียวกันได้ โดยจะมีแท็กที่บอกความแตกต่าง (unique identification tag) ในแต่การสนทนาทำให้โปรเว็บสามารถรองรับการติดต่อจากหลายไคเอนท์ได้พร้อมกัน

เมื่อได้รับการร้องขอจากไคเอนท์ โปรแกรมจะทำงานทันทีที่โปรเว็บว่างอยู่ แต่ถ้ายังไม่ว่างก็จะเข้าคิวรอจนกว่าโปรเว็บจะว่าง

3.1.3 กระบวนการทำงานของโปรเว็บเซิร์ฟเวอร์

เมื่อเรานำโปรเว็บไปทำงานบนเซิร์ฟเวอร์และในการติดต่อกับไคเอนท์ผ่านทางบราวเซอร์ โปรเว็บมีกระบวนการทำงานดั่งนี้



รูปที่ 3-1 รูปบล็อกไดอะแกรมการทำงาน

1. ไคเอนท์ร้องขอเว็บหน้าแรกผ่านทาง URL จากเซิร์ฟเวอร์
2. เมื่อไคเอนท์รับหน้าเว็บเรียบร้อยแล้ว
3. ไคเอนท์สับมิทฟอร์มโดยกดปุ่ม"Submit" บนหน้าเว็บข้อมูลต่างๆ ในฟอร์มจะถูกนำไปเติมใน URL อัตโนมัติแล้ว URL จะถูกส่งไปยัง CGI บนเซิร์ฟเวอร์
4. ถ้าโปรเว็บยังไม่ทำงานจะถูกเรียกให้ทำงาน
5. โปรเว็บจะรับ URL มาแยกเอาข้อมูลต่างๆออกมาและส่งไปยัง WIN-PROLOG และโปรแกรมที่เขียนขึ้นแต่ถ้าทั้งสองยังไม่ทำงานก็จะถูกเรียกให้ทำงาน

6. โปรแกรมจะเริ่มทำงาน ประมวลผลข้อมูลที่รับมาอย่างชาญฉลาด
7. หน้าเว็บถัดไปจะถูกสร้างขึ้นแล้วส่งไปยังไคเอนท์
8. เมื่อจบการทำงาน WIN-PROLOG ยังคงทำงานต่อไป ถ้าไม่มีการติดต่อก็จะจบการทำงานเอง
9. เมื่อไคเอนท์รับหน้าเพ็จถัดไปแล้วถ้ายังต้องมีการติดต่อกับโปรแกรมอีกก็กลับไปยังข้อ 2
10. การติดต่อสิ้นสุดลง

3.2 การติดตั้งและตั้งค่าโปรเว็บเซิร์ฟเวอร์

3.2.1 ความต้องการของระบบ

โปรแกรมประยุกต์บนโปรเว็บเซิร์ฟเวอร์สามารถทำงานได้บน Microsoft Windows 2000, Windows NT, Windows 98 หรือ Windows 95 และต้องติดตั้งเว็บเซิร์ฟเวอร์และบราวเซอร์ และต้องมี WIN-PROLOG เวอร์ชั่น 3.50 ขึ้นไป

3.2.2 เว็บเซิร์ฟเวอร์

ในปริญญานิพนธ์นี้ได้เลือกใช้ Apache HTTP Server ซึ่งเป็น open-source HTTP server สำหรับยูนิกซ์และวินโดว์เป็นเซิร์ฟเวอร์ที่ได้รับความนิยมมากที่สุดในปัจจุบันคิดเป็น 63 % ของเซิร์ฟเวอร์ทั้งหมด สามารถดาวน์โหลดได้ที่ เว็บไซต์ของ Apache

http://www.apache.org หรือ

http://httpd.apache.org/docs/windows.html

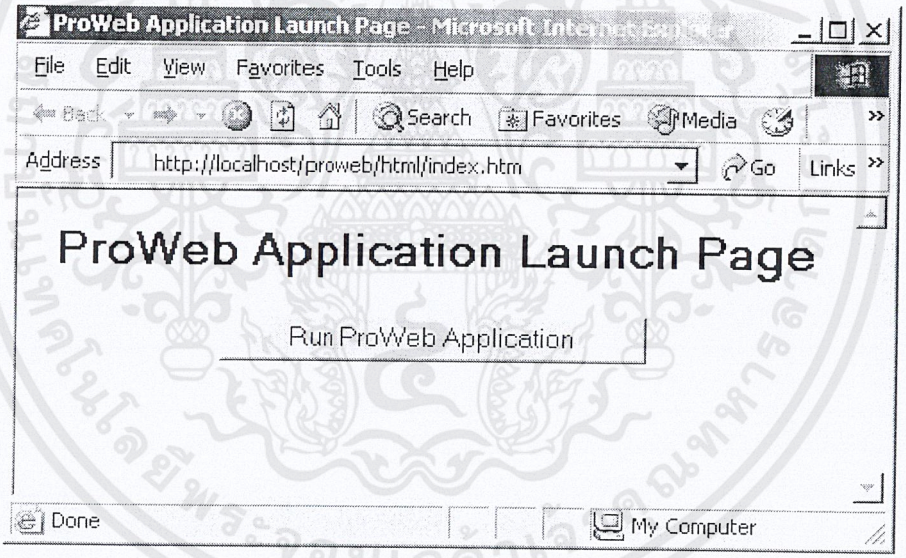
3.2.3 การติดตั้งโปรเว็บ

หลังจากติดตั้ง Apache แล้วก็ทำการเซ็ตค่า /Proweb โดยไปแก้ไขไฟล์ httpd.conf ในไดเรคทอรี่ con โดยจะต้องเพิ่มคำสั่งเหล่านี้ลงไป

```
ScriptAlias /proweb/ "C:/WIN-PROLOG4040/PROWEB/"
        ScriptAlias /ProWeb/ "C:/WIN-PROLOG4040/PROWEB/"
        <Directory "C:/WIN-PROLOG4040/PROWEB">
                AllowOverride None
                Options None
                Order allow,deny
                Allow from all
        </Directory>
```

```
Alias /map "C:/WIN-PROLOG4040/PROWEB/HTML"
<Directory "C:/WIN-PROLOG4040/PROWEB/HTML">
        Options Indexes FollowSymLinks MultiViews IncludesNoExec
        AddOutputFilter Includes html
        AllowOverride None
        Order allow,deny
        Allow from all
</Directory>
```

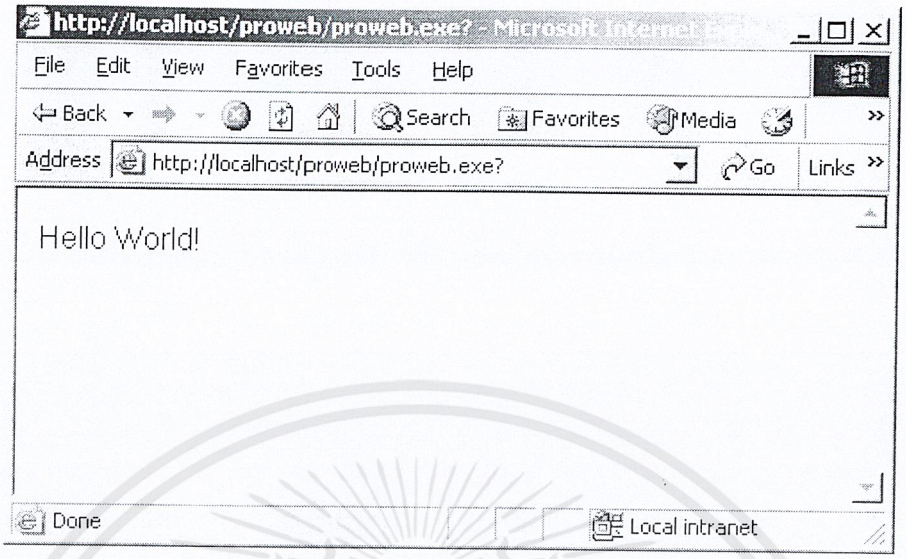
นำไฟล์นามสกุลทั้งหมดไปใส่ในไดเรคทอรี่ HTML หลังจากนั้นทดสอบการทำงานของ Apache โดยพิมพ์ URL ดังรูปที่ 3-2



รูปที่ 3-2 ทดสอบการทำงานของ Apache

3.2.4 การสร้างไฟล์ PROWEB.SYS

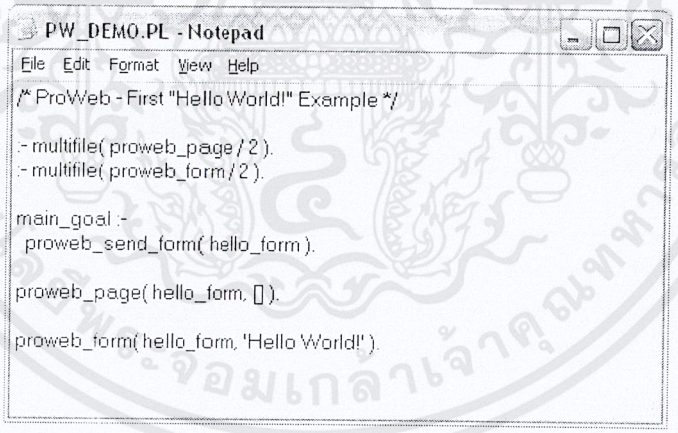
ทำสำเนาไฟล์ PRO386W.EXE จากไดเรคทอรี่ WIN-PROLOG ไปยังไดเรคทอรี่โปรเว็บแล้วเปลี่ยนชื่อเป็น PROWEB.SYS หลังจากนั้นทดลองโปรแกรมประยุกต์บนโปรเว็บโดยกดที่ "Run ProWeb Application" ผลที่ได้จะเป็นดังรูปที่ 3-3



รูปที่ 3-3 ผลลัพธ์โปรแกรม Hello World

3.3 การใช้งานโปรเว็บ

การใช้งานโปรเว็บผู้ใช้ต้องมีความเข้าใจ HTML เป็นอย่างดี จากนั้นมาดูตัวอย่างโปรแกรม Hello World



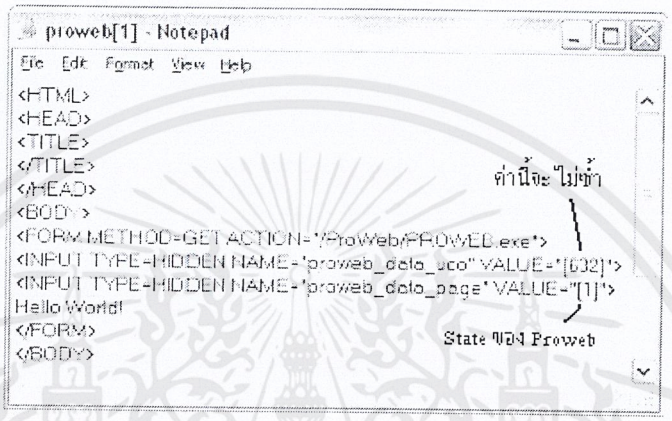
รูปที่ 3-4 รูปโค้ดโปรแกรม Hello World

ในทุกโปรแกรมบนโปรเว็บจะเริ่มทำงานที่เพรดดิเคท main_goal/0 หลังเพรดดิเคทนี้จะเป็นเพรดดิเคท proweb_send_form(hello_form)เป็นคำสั่งให้โปรเว็บนำ hello_form ไปทำให้เป็นหน้าเว็บที่สมบูรณ์ส่งไปยังไคเอนท์ แต่ก่อนที่โปรเว็บจะส่งหน้าเว็บไปยังไคเอนท์ก็จะต้องทำการสร้างโค้ด HTML ที่จำเป็นก่อน

โปรเว็บจะทำการค้นหา hello_form ที่อยู่ในตัวโปรแกรมจากการนิยามจากเพรดดิเคท proweb_form/2 โดยที่อาร์กูเมนต์ตัวแรกเป็นชื่อของฟอร์ม ตัวถัดมาเป็นการนิยามฟอร์มเหล่านั้น

โปรเว็บจะมี HTML form ที่ประกอบด้วย "Hello World" แต่ยังไม่มี HTML page ดังนั้นจึงค้นหาเพรดดิเคท proweb_page/2 ซึ่งจะนิยาม HTML page เอาไว้ดังตัวอย่างนี้ proweb_page(hello_form, [])

โค้ด HTML ที่ได้จะเป็นดังรูปที่ 3-5



รูปที่ 3-5 รูปโค้ด HTML ของโปรแกรม Hello World

พิจารณาในโค้ด HTML ที่โปรเว็บสร้างขึ้นมาจะมีแท็กที่โปรเว็บสร้างขึ้นมาเองอัตโนมัติ คือ <FORM METHOD=GET ACTION="/ProWeb/PROWEB.exe"> โปรเว็บสร้างแท็กนี้ในทุกๆฟอร์ม ส่วน <INPUT TYPE=HIDDEN NAME="proweb_data_uco" VALUE="[1027]"> และ <INPUT TYPE=HIDDEN NAME= "proweb_data_page" VALUE="[1]"> โปรเว็บสร้างขึ้นเพื่อไว้ทำประโยชน์ในการทำย้อนกลับและอื่นๆ

3.4 โปรดาต้า

โปรดาต้า(ProData) เป็นตัวเชื่อมระหว่าง LPA-PROLOG for Windows กับ Database Management System (DBMSs) ทุกๆตัวโดยผ่าน Open Database Connectivity (ODBC) ทำให้โปรดาต้าไม่ขึ้นอยู่กับฐานข้อมูลที่ใช้ และทำให้มีความยืดหยุ่นในการทำงานในบางเพรดดิเคท

โปรดาต้าอนุญาตให้ตารางในฐานข้อมูลสามารถถูกเรียกใดจากโปรล็อกเหมือนกับการใช้งาน fact ในโปรล็อก ความง่ายนี้ทำให้สามารถใช้กฎในโปรล็อกกับฐานข้อมูลได้ Backtracking , cut , fail, not และ กระบวนการทำงานมาตรฐานของโปรล็อกสามารถทำงานเข้าถึงฐานข้อมูลภายนอกได้

ความสามารถของโปรเว็บ

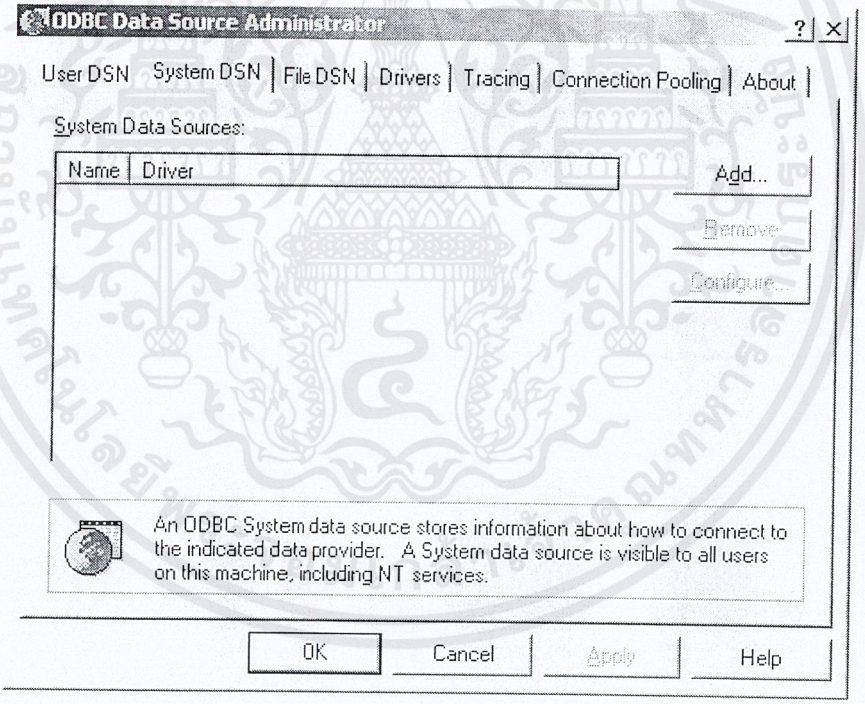
- สามารถทำ Concurrent ได้ควบคุมโดยโปรแกรม
- สามารถทำคำสั่ง SQL จากโปรล็อกได้แต่ขึ้นอยู่กับไดร์เวอร์ของ ODBC ด้วย

- สามารถแสดงข้อผิดพลาดจากระบบฐานข้อมูลได้
- แปลงข้อมูลอัตโนมัติ
- สามารถเข้าถึง Data Dictionary แต่ขึ้นอยู่กับ DBMS ด้วย
- ซ่อน Cursor manipulation โดยที่ไม่ต้องเขียนโปรแกรมควบคุม
- สนับสนุนทุกๆ DBMS ที่มีไดร์เวอร์ใน ODBC
- สามารถเชื่อมต่อได้หลาย DBMS และทำการ join ข้ามระบบฐานข้อมูลกันได้

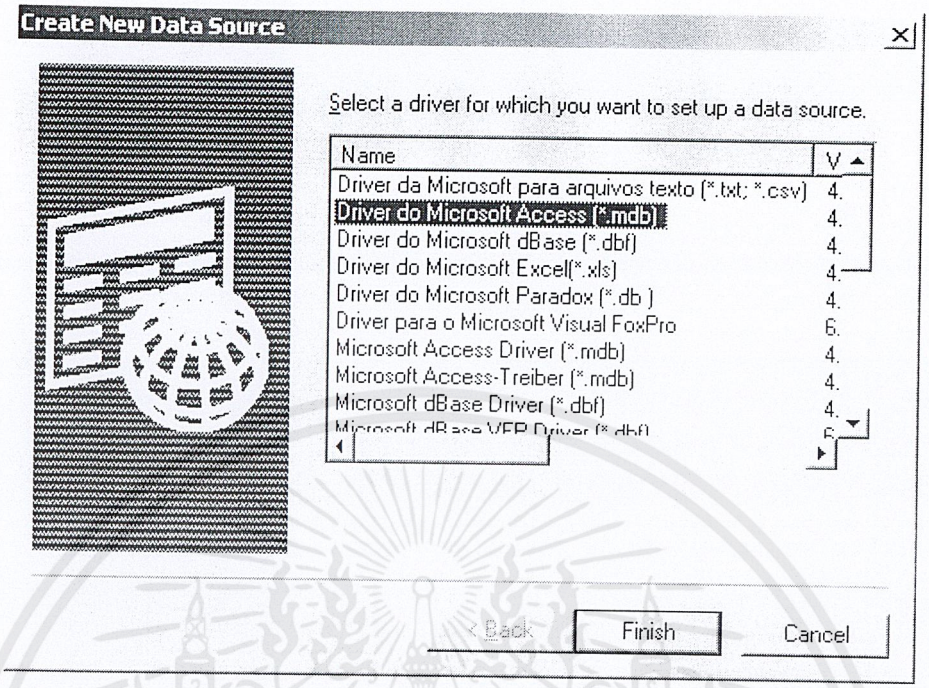
3.5 การติดตั้งโปรดาต้า

หลังจากติดตั้ง WIN-PROLOG แล้วจะได้ไฟล์ 2 ไฟล์คือ DBLINK.PC ไฟล์ที่ใช้ติดต่อกับโปรล็อกและ LPADBW.DLL ใช้ติดต่อกับไดร์เวอร์ของ ODBC

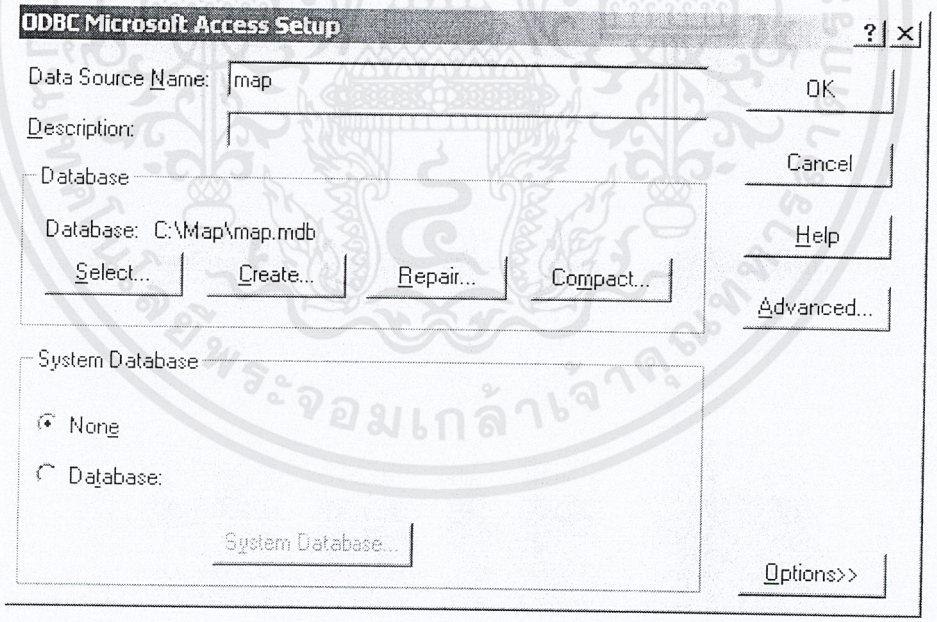
สำหรับการติดต่อในแต่ละ DBMS นั้นจะผ่านไดร์เวอร์ของ ODBC ที่ติดตั้งเรียบร้อยแล้วและต้องเซ็ตค่า Data Source ใน ODBC Administration ใน Control Panel ดังในรูป 3-6, 3-7 และ 3-8



รูปที่ 3-6 ODBC Data Sources Administrator



รูปที่ 3-7 เลือก Driver สำหรับการสร้าง Data Source



รูปที่ 3-8 การสร้าง Data source โดยใช้ Access

3.6 การใช้งานโปรดาต้า

3.6.1 การเชื่อมโปรล็อกกับโปรดาต้า

การเชื่อมโปรดาต้าจำเป็นต้องมีไฟล์ DBLINK.PC และ LPADBW.DLL โดยสามารถโหลดจาก WIN-PROLOG ด้วยคำสั่งนี้

```
?- ensure_loaded( system(dblink) ).<return>
yes
```

3.6.2 การเชื่อมกับฐานข้อมูล

หลังจากโหลดโปรดาต้าเรียบร้อยแล้วก็สามารถทำการเชื่อมต่อกับฐานข้อมูลได้ด้วยเพรดดิเคท db_connect/1 โดยที่อาร์กูเมนต์ตัวแรกเป็นชื่อ Data Source

```
?- db_connect( map ). <return>
yes
```

เราสามารถเพิ่มชื่อ Data Source ด้วย 'attribute=value' โดยที่ attribute และ value ขึ้นอยู่กับไดร์เวอร์ ODBC ที่ใช้ด้วยเช่นถ้ามีรหัสผ่านเป็น mapservice ก็จะต้องทำดั่งนี้

```
?- db_connect( 'map;PWD=mapservice' ). <return>
yes
```

หลังจากการใช้งานเรียบร้อยแล้วก็สามารถหยุดการเชื่อมต่อด้วย เพรดดิเคท db_disconnect/0 ถ้าออกจากโปรแกรมแล้วไม่ทำการหยุดการเชื่อมต่อโปรดาต้าก็จะหยุดการเชื่อมต่อให้อัตโนมัติ

3.6.3 การใช้งานตารางในฐานข้อมูล

เราสามารถใช้งานข้อมูลในตารางที่ต้องการได้ง่ายๆ โดยการ attach เข้าเป็นเพรดดิเคทในโปรล็อกได้ด้วยเพรดดิเคท

```
?- db_attach( dept, 'DEPT' ). <return>
yes
?- dept( Deptno, Dname, Loc ).<return>
Deptno = 10,
Dname = 'ACCOUTING',
Loc = 'New York'
```

เพรดดิเคท db_tuple/2 ก็เป็นอีกเพรดดิเคทหนึ่งที่สามารถคิวรี่ข้อมูลได้โดยอาร์กูเมนต์ตัวแรกเป็นชื่อตารางและตัวที่สองเป็นลิสต์ข้อมูลในหนึ่งแถว

```
?- db_tuple( 'Shippers', R ). <return>
```

R = [1, 'Speedy Express', '(503) 555-9831']

3.6.4 การใช้คำสั่งคิวรี่โดยตรง

การใช้คำสั่งคิวรี่โดยตรงได้จากเพรดดิเคท db_sql_select/3

db_sql_select(HDBC, SQL, L) โดยที่

HDBC เป็นอะตอม (atom)

SQL เป็นคำสั่งคิวรี่เช่น 'select ename, empno from emp'

L เป็นลิสต์ของข้อมูลในแถว

บทที่ 4
ระบบแผนที่และการวินิจฉัย

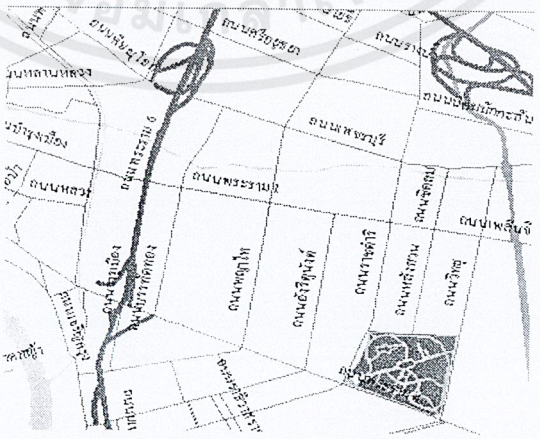
4.1 ระบบแผนที่ (Map System)

แผนที่ที่ใช้อยู่ในปัจจุบันแบ่งได้เป็น 2 ชนิดใหญ่ ๆ[2] คือแผนที่เฉพาะเรื่อง (Thematic map) และแผนที่ภูมิประเทศ (Topographic map) โดยที่แผนที่เฉพาะเรื่องนี้เป็นแผนที่ที่มีองค์ประกอบอื่นๆเข้ามามาก ส่วนแผนที่ภูมิประเทศจะเป็นแผนที่ที่เน้นแสดงสภาพทางภูมิศาสตร์โดยเฉพาะ ในวิทยานิพนธ์นี้จะขออธิบายเฉพาะแผนที่เฉพาะเรื่องที่จะนำมาใช้ในวิทยานิพนธ์ ดังต่อไปนี้

แผนที่เฉพาะเรื่อง [2] คือแผนที่ที่แสดงรายละเอียดของข้อมูลเชิงพื้นที่ที่ต้องการนำเสนอโดยการแปลงข้อมูลเหล่านั้นให้เป็นเครื่องหมายแผนที่เสียก่อน แล้วนำไปพิมพ์ซ้อนทับลงบนแผนที่ฐาน ตามตำแหน่งที่ตั้งของข้อมูลนั้นๆ ซึ่งหมายถึงประกอบไปด้วย ข้อมูลเชิงพื้นที่ (Spatial Data) และแผนที่ฐาน (Base map)ในการทำแผนที่นี้เมื่อเตรียมการเสร็จแล้วจะทำการพิมพ์ลงบนกระดาษ (Paper map) สำหรับปัญหาของแผนที่แบบกระดาษคือ ถ้ามีการเพิ่มเติมหรือแก้ไขข้อมูลจะไม่สามารถแก้ไขข้อมูลในเวลาสั้นๆได้ จะต้องทำการพิมพ์แผนที่ออกมาใหม่ทั้งหมด ทำให้เสียเวลาและค่าใช้จ่ายมาก ปัจจุบันนี้ได้ทำการดัดแปลงแผนที่เฉพาะเรื่อง มาจัดเก็บไว้ในคอมพิวเตอร์ โดยนำข้อมูลไปเก็บไว้ในฐานข้อมูลคอมพิวเตอร์มีการแสดงผลโดยการวางซ้อนทับฐานข้อมูล การนำคอมพิวเตอร์เข้ามาใช้ในการจัดเก็บข้อมูลนี้จะทำได้ดีกว่าแผนที่กระดาษ เพราะว่าสามารถเลือกดูชั้นข้อมูลที่จำเป็นเท่านั้น ทำให้เข้าใจง่ายกว่าแผนที่กระดาษ

4.2 แผนที่ดิจิตอล

แผนที่ดิจิตอล (Digital map) หรือแผนที่เชิงตัวเลข เป็นแผนที่ที่ใช้คอมพิวเตอร์ในการประมวลผล และมีการจัดเก็บข้อมูลของแผนที่ให้อยู่ในรูปของข้อมูลคอมพิวเตอร์ ซึ่งข้อมูลคอมพิวเตอร์จะทำการจัดเก็บในรูปแบบของฐานข้อมูลคอมพิวเตอร์ ตัวอย่างของแผนที่ดิจิตอลแสดงในรูปที่ 4-1 แผนที่ดิจิตอลแบ่งตามการจัดเก็บออกเป็น 2 แบบคือ แบบราสเตอร์และแบบเวกเตอร์

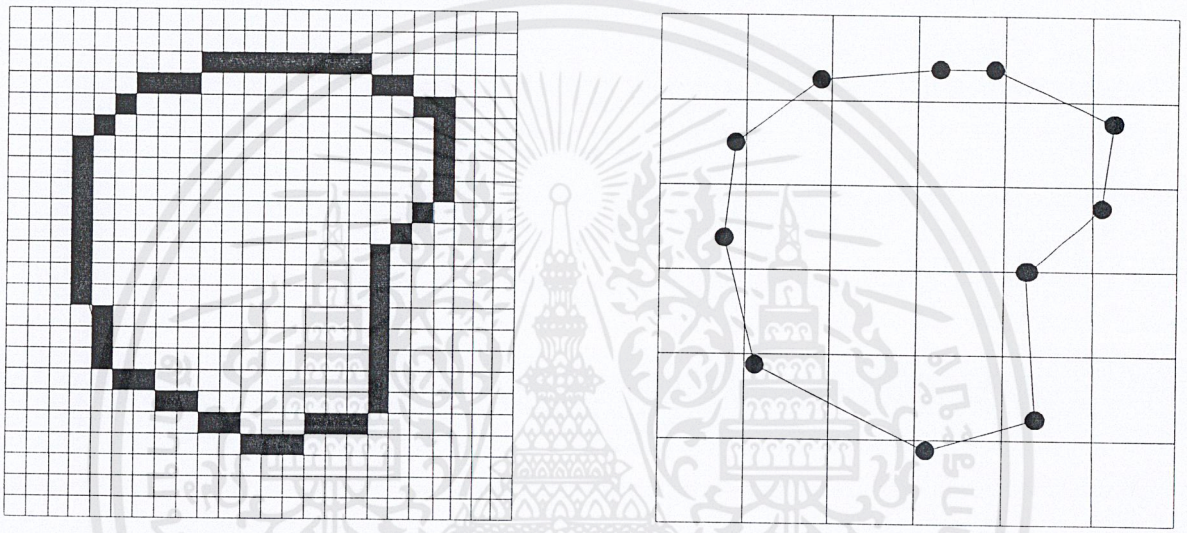


รูปที่ 4-1 ตัวอย่างแผนที่ดิจิตอล

แผนที่แบบราสเตอร์ หมายถึงแผนที่ที่มีการจัดเก็บและแสดงผลในรูปของจุดภาพ การสร้างแผนที่แบบนี้จะได้โดยรับภาพแผนที่จากแผนที่กระดาษผ่านทางเครื่องสแกนภาพซึ่งวิธีการสแกนภาพเป็นการนำรูปภาพทั้งรูปเข้าไปเก็บไว้ในลักษณะของรูปภาพ ซึ่งการแก้ไขจะทำได้ยากรวมทั้งใช้เนื้อที่ในการจัดเก็บมาก

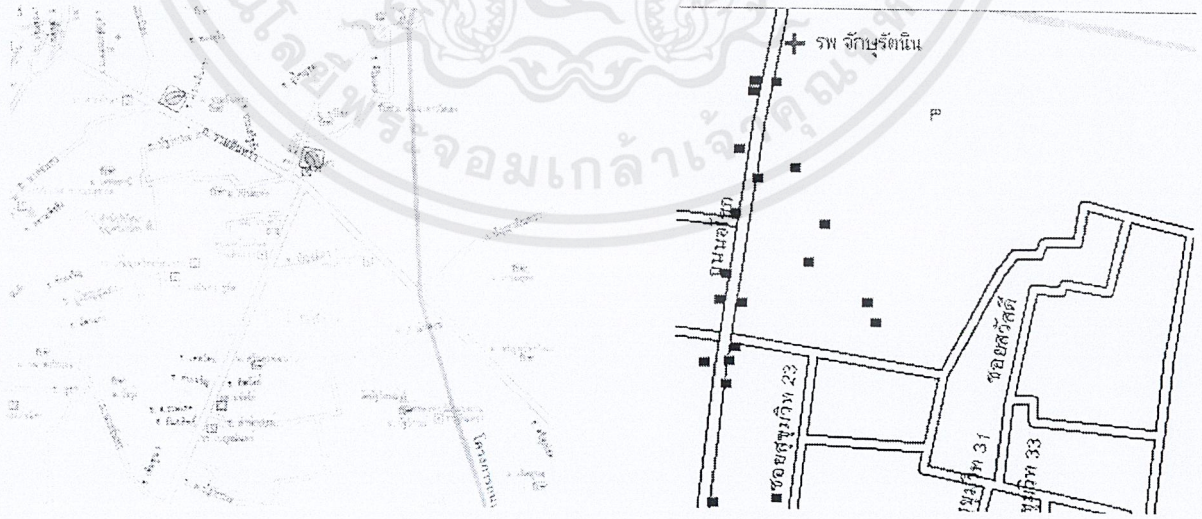
แผนที่แบบเวกเตอร์ หมายถึงแผนที่ที่มีการจัดเก็บและแสดงผลในรูปของลายเส้น และมีทิศทาง การสร้างแผนที่แบบนี้ทำได้โดยใช้วิธีการลอกแบบจากเครื่องดิจิตไตเซอร์ ซึ่งจะเก็บเฉพาะข้อมูลในส่วนที่ต้องการลอกแบบ ดังนั้นข้อมูลแบบนี้จึงใช้เนื้อที่ในการจัดเก็บน้อยกว่า สามารถแก้ไขได้ในภายหลังโดยที่มาตรส่วนไม่ผิดไปจากเดิม

รูปแบบการแสดงผลแบบราสเตอร์ และเวกเตอร์แสดงในรูปที่ 4-2(ก) และ 4-2(ข) ตามลำดับและตัวอย่างของแผนที่แบบราสเตอร์และเวกเตอร์แสดงในรูปที่ 4-3(ก) และ 4-3(ข) ตามลำดับ



(ก) (ข)

รูปที่ 4-2 การแสดงผล แบบราสเตอร์ และแบบเวคเตอร์



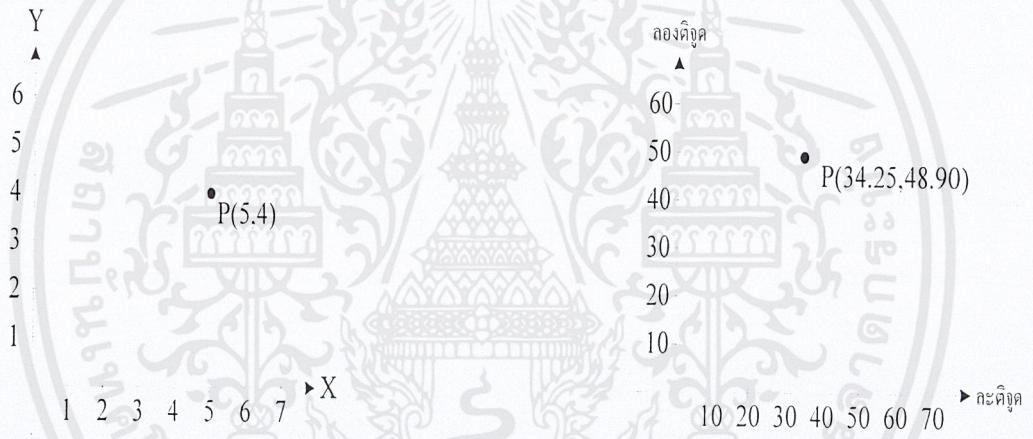
(ก) (ข)

รูปที่ 4-3 ตัวอย่างของแผนที่แบบราสเตอร์ และแบบเวกเตอร์

สำหรับในวิทยานิพนธ์ได้เลือกใช้แผนที่ตัวเมืองกรุงเทพฯ มาตราส่วน 1:20,000 ซึ่งเป็นประเภทมาตราส่วนใหญ่สามารถแสดงรายละเอียดต่างๆได้มากและชัดเจน เช่น ถนน สถานที่สำคัญ เป็นต้น ซึ่งเป็นข้อมูลแบบเวกเตอร์โดยจะใช้ลักษณะของจุดและเส้น ในการแสดงลักษณะทางภูมิศาสตร์ซึ่งขบวนการของข้อมูลแบบเวกเตอร์นี้จะใช้คู่ขอพิกัด X,Y เป็นตัวชี้ตำแหน่งและลักษณะของสิ่งต่าง ๆ

4.3 ระบบพิกัดบนแผนที่

ระบบพิกัดบนแผนที่จะมีการอ้างอิงพิกันที่เหมือนกับระบบพิกัดฉากในทางเลขาคณิต ที่ประกอบไปด้วยแกน X และแกน Yโดยจุดกำเนิดหมายถึงจุดตักระหว่างแกน X และแกน Y เมื่อแทนด้วยระบบพิกัดบนแผนที่แล้วแกน X จะหมายถึงเส้นละติจูด และแกน Y จะหมายถึงเส้นลองติจูด เมื่อพิจารณาระบบพิกัดบนโลกแล้วเราจะพิจารณาเป็นลักษณะของ 3 มิติคือ X, Y, Z โดย Z จะหมายถึงค่าความสูง ระบบนี้จะใช้ในการอ้างอิงในระบบ GPS เป็นหลักสำหรับในวิทยานิพนธ์นี้เราจะพิจารณาเฉพาะเส้นละติจูดและลองติจูดเป็นหลักรูปที่ 4-4 เป็นการเปรียบเทียบให้เห็นระหว่างพิกัดในทางเลขาคณิตกับพิกัดบนแผนที่



รูปที่ 4-4 ระบบพิกัด

4.3.1 การคำนวณระยะทางบนแผนที่

จากที่ได้อธิบายเกี่ยวกับพิกัดตำแหน่งไปแล้ว จะพบว่าทุกพื้นที่บนแผนที่จะประกอบได้พิกัดตำแหน่งมากมาย ดังนั้นการคำนวณระยะทางจึงหมายถึงระยะห่างระหว่างตำแหน่ง 2 ตำแหน่งบนแผนที่ ซึ่งเมื่อเราทราบระบบพิกัดตำแหน่งบนแผนที่แล้ว เราสามารถหาระยะทางบนแผนที่ได้

4.3.2 ระยะทางระหว่างจุดสองจุดบนแผนที่

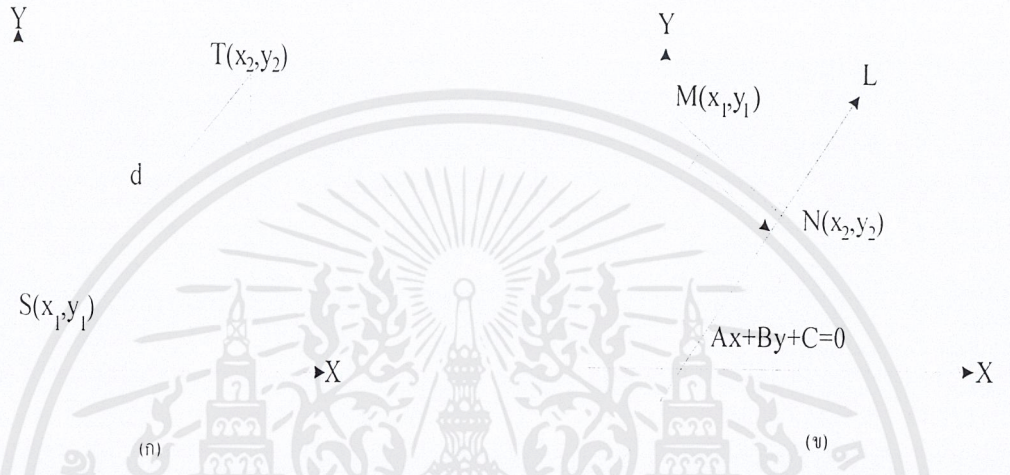
ระยะทางระหว่างจุดสองจุดบนแผนที่คำนวณได้ตาม (4.1) โดยที่ d หมายถึงระยะทางระหว่างตำแหน่งทั้ง 2 และ (x, y) คือ พิกัดตำแหน่งใด ๆ

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \qquad (4.1)$$

4.3.3 ระยะทางระหว่างจุดถึงเส้นตรง

ระยะทางระหว่างจุดถึงเส้นตรง คำนวณได้ตาม (4.2) โดยที่ d หมายถึงระยะทางระหว่างตำแหน่งทั้ง 2 และ (x, y) คือพิกัดตำแหน่งใด ๆ

$$d = \frac{|Ax1 + By1 + C|}{\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}} \qquad (4.2)$$



รูปที่ 4-5 การคำนวณระยะห่างระหว่างจุดและเส้นตรงกับจุด

ตามรูปที่ 4-5(ก) เป็นตัวอย่างการคำนวณระยะทางจากตำแหน่ง S(x1, y1) ถึงตำแหน่ง T(x2, y2) ค่าของระยะทาง d คำนวณได้จาก (1) และรูปที่ 4-5(ข) เป็นตัวอย่างการคำนวณระยะทางจากตำแหน่ง M(x1, y1) ไปยังตำแหน่ง N(x2, y2) ที่อยู่บนเส้นตรง Ax + By + C = 0

จากหลักการนี้เราจะนำไปใช้ในการคำนวณหาพื้นที่บนแผนที่ เช่น พื้นที่สี่เหลี่ยมผืนผ้าคำนวณได้จาก d1*d2 เมื่อ d1 คือความกว้างและ d2 คือความยาว

4.4 ส่วนการวินิจฉัย

ส่วนการวินิจฉัยนี้มีหน้าที่หลัก คือการค้นหาเส้นทางบนแผนที่และการวางแผนเส้น รวมทั้งการวางแผนใหม่ ในส่วนการประยุกต์ใช้งานเป็นระบบนำร่อง แต่ก่อนที่จะกล่าวถึง การค้นหาเส้นทางบนแผนที่และการวางแผนเส้นทาง จะขออธิบายเกี่ยวกับหลักการค้นหาทั่วๆไป และอธิบายการประยุกต์ใช้หลักการค้นหาที่นำมาใช้ในระบบแผนที่ในลำดับต่อไป สำหรับเทคนิคของการค้นหาแบ่งได้เป็น 2 ประเภทใหญ่ๆ คือ การค้นหาประเภท Uniformed State Space Search และ การค้นหาประเภท Informed State Space Search ที่จะได้อธิบายในหัวข้อต่อไป

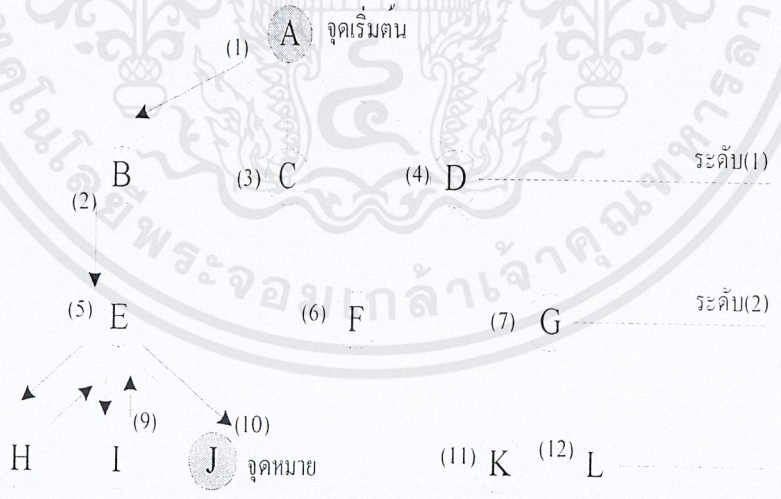
4.5 การค้นหาประเภท Uniformed State Space Search

การค้นหาประเภทนี้เป็นการค้นหาข้อมูลโดยการพิจารณาทางเลือกเพื่อไปให้ถึงจุดหมาย ในการค้นหาแบบนี้จะไม่มีข้อมูลใด ๆ มาช่วยในการพิจารณาหรือสนับสนุนการเลือกเส้นทางที่เดินทางไป แต่จะใช้หลักการกำหนดทิศทางในการเลือกเส้นทางที่จะค้นหาไปยังแนวราบหรือแนวลึก จนกว่าจะเจอจุดหมาย ซึ่งหลักการนี้ไม่รับประกันว่าจะค้นหาเป้าหมายเจอหรือไม่เมื่อเทียบกับวิธีการอื่น ทั้งนี้เนื่องจากขาดข้อมูลในการพิจารณา การค้นหาแบบนี้จึงแบ่งออกเป็น 2 วิธีด้วยกันคือ การค้นหาตามแนวลึกก่อน และการค้นหาตามแนวกว้างก่อน

4.5.1 การค้นหาตามแนวลึกก่อน (Depth First Search)

วิธีการค้นหาแบบแนวลึกก่อน เป็นการเลือกทิศทางที่จะเดินทางไปยังโหนดจุดหมายในแนวลึก หรือแนวดิ่งก่อน โดยการเลือกทิศทางเดินทางไปยังโหนดที่อยู่ซ้ายสุดก่อน จากนั้นจะทำการเลื่อนต่อไปยังโหมดที่อยู่ด้านใต้อีกหนึ่งระดับทำเช่นนี้จนกว่าจะเจอจุดหมายเมื่อเจอโหนดล่างสุด(ยังไม่ใช่จุดหมาย)ก็จะย้อนกลับ (Backtrack) ขึ้นมากหนึ่งระดับเพื่อพิจารณาทางด้านขวาต่อไป ทำอย่างนี้ต่อไปนกว่าจะเจอโหนดจุดหมาย ดังรูปที่ 4-6

ตามรูปที่ 4-6 เริ่มต้นเดินทางที่โหนด A (หมายเลข 1) เมื่อเดินทางตามแนวลึกระดับที่ 1 และทำการพิจารณาโหนดซ้ายสุดก่อนซึ่งจะพบกันโหนด B (หมายเลข 2) เมื่อตรวจสอบแล้วไม่ใช่ตำแหน่งจุดหมายจึงเดินทางต่อไปตามแนวลึกระดับที่ 2 ทางซ้ายสุดก่อนและพบกับโหนด E (หมายเลข 5) เมื่อตรวจสอบแล้วยังไม่พบจุดหมายอีกจึงเริ่มเดินทางต่อไปตามแนวลึกระดับที่ 3 อีกหนึ่งระดับซ้ายสุดก่อนจะพบโหนด H(หมายเลข 8) ตรวจสอบแล้วยังไม่พบจุดหมาย เมื่อเริ่มเดินทางต่อ

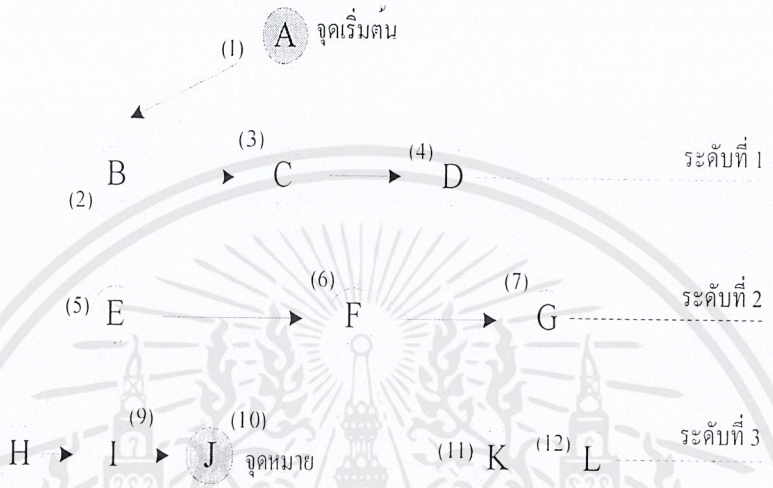


รูปที่ 4-6 การค้นหาแนวลึก

ตามแนวลึกหนึ่งระดับแต่เนื่องจากที่โหนดนี้ไม่มีโหนดต่อในแนวลึกจึงต้องทำการย้อนกลับขึ้นมาก 1 ระดับ คือ โหนด E (หมายเลข 5) ทำเช่นนี้จนกระทั่งค้นพบโหนดจุดหมายคือ J ดังนั้น เส้นทางทั้งหมดตั้งแต่เริ่มต้นเดินทางจนถึงจุดหมายตามหมายเลข ดังนี้ "1→2→5→8→5→9→5→10" ดังนั้นเส้นทางที่ถูกเลือกผ่านโหนดต่าง ๆ ดังนี้ A, B, E, H, I และ J

4.5.2 การค้นหาตามแนวกว้างก่อน (Breadth First Search)

วิธีการค้นแบบแนวกว้าง เป็นการเลือกทิศทางที่จะเดินทางไปยังโหนดจุดหมายในแนวกว้างหรือแนวราบก่อน คือจากโหนดเริ่มต้นค้นหาตามแนวกว้างก่อนแล้วตรวจสอบว่าใช่จุดหมายหรือไม่ ถ้าใช่จะหยุดค้นหา ถ้าไม่ใช่จะค้นหาต่อไปจนกว่าจะพบจุดหมาย ดังรูปที่ 4-7



รูปที่ 4-7 การค้นหาแนวกว้าง

ตามรูปที่ 4-7 เริ่มต้นเดินทางจากโหนด A(หมายเลข 1) เมื่อเดินทางตามแนวกว้างระดับที่ 1 จะพบกับโหนด B (หมายเลข 2), C (หมายเลข 3)และ D (หมายเลข 4) ตามลำดับ ซึ่งตรวจสอบแล้วยังไม่ใช่จุดหมาย จากนั้นจะทำการค้นหาอีก 1 ระดับ โดยพลโหนด E (หมายเลข 5), F (หมายเลข 6) , และ G (หมายเลข 7) ตามลำดับ ซึ่งเมื่อตรวจสอบแล้วยังไม่ใช่จุดหมายอีก จึงค้นหาอีก 1 ระดับ จนพบตำแหน่งจุดหมาย คือ J เส้นทางทั้งหมดตั้งแต่เริ่มต้นเดินทางจนถึงจุดหมายตามหมายเลขดังนี้ "1→2→3→4→5→6→7→8→9→10" ดังนั้นเส้นทางที่ถูกเลือกผ่านโหนดต่าง ๆ คือ A, B, C, D, E, F, G, H, I และ J ตามลำดับ

จากหลักการค้นหาตามแนวลึกก่อน และการค้นหาตามแนวกว้าง จะพบว่ามีโหนดมาก ๆ จะทำให้ใช้เวลาในการค้นหานานมากจนกว่าจะพบจุดหมาย และใช้พื้นที่หน่วยความจำมากเกินไป ทั้งนี้เนื่องจากวิธีการค้นหาดังกล่าวเป็นการค้นหาแบบไม่มีสิ่งนำทาง ดังนั้นวิธีการค้นหาแบบ Informed State Space Search จึงถูกนำมาพิจารณาใช้ โดยการพิจารณาเลือกข้อมูลเพื่อช่วยในการตัดสินใจเลือกทางเดินที่มีโอกาสจะเจอจุดหมายได้ง่ายกว่า ซึ่งรายละเอียดจะกล่าวถึงในขั้นตอนต่อไป

4.6 การค้นหาประเภท Informed State Space Search (Heuristic Search)

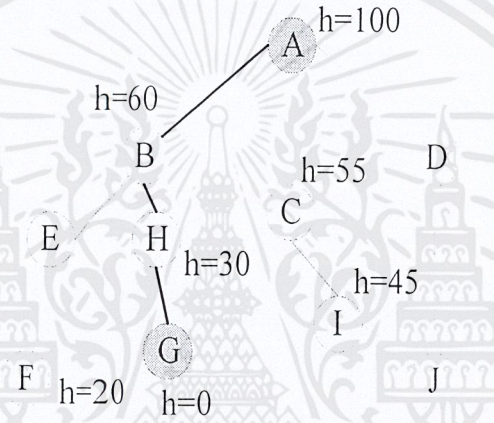
การค้นหาประเภทนี้เรียกอีกอย่างหนึ่งว่า "การค้นหาแบบฮิวริสติก" ในการค้นหานี้จะใช้ค่าฮิวริสติกฟังก์ชั่น (Heuristic Function) มาช่วยในการค้นหา ซึ่งฟังก์ชั่นนี้จะทำหน้าที่วัดขนาดของความเป็นไปได้ที่จะเดินทางไปสู่จุดหมายอาจจะเป็นตัวเลข น้ำหนัก เวลา หรือระยะทาง การค้นหาแบบนี้จะถูกเรียกว่าการค้นหาแบบดี

ที่สุดก่อน (Best First Search) ซึ่งเป็นการรวมเอาข้อดีของการค้นหาแบบลึก และการค้นหาแบบกว้าง ตัวอย่างของการค้นหาประเภทนี้ได้แก่ การค้นหาแบบกรีดี และการค้นหาแบบฮิริสติกแบบเอสตาร์

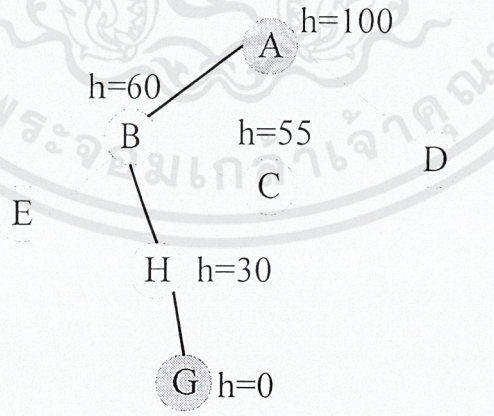
4.6.1 การค้นหาแบบกรีดี (Greedy Search)

การค้นหาแบบกรีดี เป็นการค้นหาแบบดีที่สุดประเภทหนึ่งโดยเป็นการค้นหาแบบที่เลือกค่าที่น้อยที่สุดจากโหนดใดๆไปยังโหนดจุดหมาย วิธีการค้นหาแบบนี้จะอาศัยข้อมูลมาช่วยในการพิจารณาเลือกเส้นทางโดยใช้ฟังก์ชั่นที่เรียกว่า h(n)โดย h(n) หมายถึงระยะทางจากโหนดปัจจุบันถึงโหนดจุดหมาย การค้นหาแบบนี้จะเลือกเดินทางที่มีค่าใช้จ่ายน้อยกว่าเสมอ ดังรูปที่ 4-8

ตามรูปที่ 4-8 เป็นตัวอย่างของการกำหนด h(n) ของแต่ละโหนด โดยจะเริ่มเดินทางจากโหนดเริ่มต้น (โหนด A) ไปยังโหนดจุดหมาย (โหนด G)



รูปที่ 4-8 การค้นหาแบบกรีดี



รูปที่ 4-9 การเลือกเส้นทางโดยการพิจารณาจาก h(n)

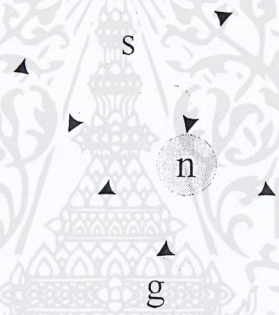
ตามรูปที่ 4-9 เป็นการอธิบายถึงการเลือกเส้นทางโดยการพิจารณาจากค่า h(n) ที่น้อยที่สุดก่อนดังนี้ เริ่มต้นเดินทางที่โหนด A เมื่อเริ่มเดินทางพบว่ามีโหนดให้เลือกอยู่ 2 โหนดคือ โหนด B และ D โดยที่โหนด B จะถูกเลือกไว้ก่อนเนื่องจากมีค่าใช้จ่ายน้อยกว่าคือ "60" จากนั้นเมื่อพิจารณาโหนด B มีโหนดที่เชื่อมต่อให้เลือก 3

โหนดคือ โหนด E, H และ C โหนด H จะถูกเลือกก่อนเพราะมีค่าใช้จ่ายน้อยกว่าคือ "30" จากนั้นจะเดินทางต่อไปจนพบจุดหมายคือ G เส้นทางทั้งหมดตั้งแต่เริ่มต้นเดินทางจนถึงจุดหมาย คือ A, B, H และ G ตามแนวเส้นทึบ

สำหรับการค้นหาแบบกรีดี นี้เส้นทางที่ถูกเลือกจึงไม่รับประกันว่าเป็นเส้นทางที่สั้นที่สุด นั่นคือถ้าโหนดใด ๆ อยู่ใกล้จุดหมาย แต่เส้นทางนั้นอาจมีระยะทางรวมมากกว่าก็ได้ ดังนั้นค่าของ h(n) เพียงอย่างเดียวจึงไม่เพียงพอในการตัดสินใจเลือกเส้นทางได้เหมาะสมที่สุด วิธีการค้นหาแบบฮิวริสติกแบบเอสตาร์จึงถูกนำมาพิจารณาในการเลือกเส้นทางที่ดีที่สุด

4.6.2 การค้าหาแบบฮิวริสติกแบบเอสตาร์

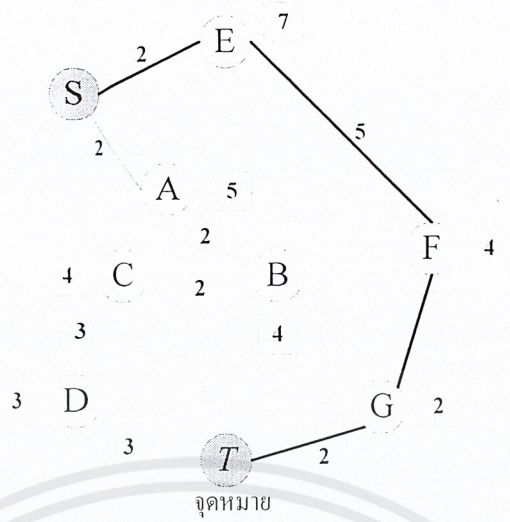
เป็นการค้นหาที่นำเอาค่าฮิวริสติกฟังก์ชั่น f(n) ที่มีค่าเท่ากับ $f(n) = g(n) + h(n)$ มาใช้ในการตัดสินใจเลือกเส้นทางที่เหมาะสมที่สุดโดย g(n) หมายถึงค่าใช้จ่ายจริงที่ใช้ในการเดินทางจากโหนดเริ่มต้น S จนไปถึงโหนดปัจจุบัน n (ซึ่งค่าใช้จ่ายในที่นี้อาจเป็นระยะทาง เวลา ฯลฯ) และค่า h(n) หมายถึงค่าใช้จ่ายประมาณการที่ใช้ในการเดินทางจาดโหนดปัจจุบัน n ไปถึงโหนดจุดหมายตามรูป 4-10



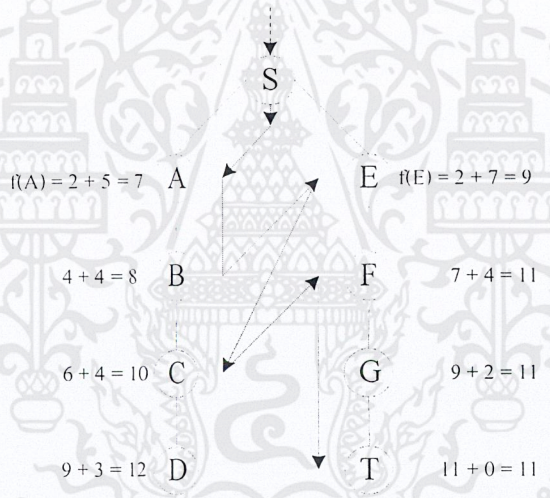
รูปที่ 4-10 ส่วนประกอบของ heuristic function f(n)

อัลกอริธึมในการค้นหาแบบเอสตาร์จะใช้ f(n) ในการตัดสินเลือกโหนดของกราฟเริ่มจากโหนดเริ่มต้น s เพื่อค้นหาโหนดลูกในแต่ละขั้น ซึ่งแต่ละขั้นจะทำให้เกิดเส้นทางที่ยังค้นหาไม่จบ เส้นทางเหล่านี้มีโหนดปลายซึ่งแต่ละอันถือว่าเป็นโหนด n ของเส้นทางนั้น ๆ ถ้าเส้นทางใดในบรรดาเส้นทางเหล่านี้ให้ค่า f(n) ของโหนด n ของตนเป็นค่าที่น้อยที่สุดในกลุ่มโหนด n อื่นๆ อัลกอริธึมจะเลือกขยายขอบเขตการค้นหาต่อในที่เส้นทางนั้น โดยขยายการค้นหาที่โหนด n ของเส้นที่เลือกลงไปยังโหนดลูกของมัน ถ้ามีโหนดลูกตัวใดปรากฏในเส้นทางที่กำลังค้นหาค้างอยู่ก็จะถูกตัวทิ้งไป การค้นหาโดยเอสตาร์จะทำทีละขั้นเช่นนี้ซ้ำแล้วซ้ำอีกไปเรื่อยๆจนกระทั้งพบโหนดจุดหมาย g เส้นทางที่เกิดขึ้นที่เชื่อมจาก s ไป g จะถือว่าเป็นเส้นทางที่ดีที่สุดที่เลือกโดยวิธีเอสตาร์

วิธีเอสตาร์จะสามารถเลือกเส้นทางที่ดีที่สุดได้ก็ต่อเมื่อ ฟังก์ชั่น g(n) ซึ่งเป็นการคิดค่าใช้จ่ายจากโหนดเริ่มต้น s ถึงโหนดปัจจุบัน n จะต้องเป็นฟังก์ชั่นที่คิดค่าออกมาแล้วไม่ต่ำกว่าค่าใช้จ่ายจริงของเส้นทางที่สั้นที่สุดที่เชื่อมจาก s ถึง n ส่วน h(n) ซึ่งเป็นการประมาณการค่าใช้จ่ายจากโหนดปลาย n ถึงโหนดจุดหมาย g จะต้องเป็นการประมาณค่าที่ไม่เกินจากค่าใช้จ่ายจริงของเส้นทางที่สั้นที่สุดที่เชื่อมจาก n ถึง g



รูปที่ 4-11 การค้นหาแบบฮิวริสติก



รูปที่ 4-12 ลำดับการค้นหาแบบฮิวริสติก

ตามรูปที่ 4-11 แสดงการค้นหาแบบฮิวริสติก เป็นการเดินทางจากโหนด s ไปยังโหนด T เมื่อเริ่มต้นเดินทาง มีโหนดให้เลือกอยู่ 2 โหนดคือ A กับ E เส้นทางที่ถูกเลือกคือ A ที่มีค่าใช้จ่ายเท่ากับ 7 เมื่อเดินทางต่อไปจะพบว่าโหนด B มีค่าใช้จ่ายเท่ากับ 8 จากนั้นเดินทางต่อไปจะพบโหนด C ที่มีค่าใช้จ่ายเท่ากับ 10 ที่โหนดนี้จะพบว่าค่าใช้จ่ายมากกว่า โหนด E ที่มีโหนดเท่ากับ 9 ดังนั้นจึงต้องย้อนกลับไปเลือกโหนด E เมื่อเริ่มเดินทางต่อไปพบว่าโหนด C มีค่าใช้จ่ายน้อยกว่าโหนด F จึงเลือกโหนด C เมื่อเดินทางต่อไปพบว่าโหนด F มีค่าใช้จ่ายน้อยกว่าคือ 11 จากนั้นเริ่มเดินทางต่อไปก็จะเลือกเส้นทางที่มีค่าใช้จ่ายน้อยกว่าเป็นหลักจนพบจุดหมาย ดังนั้นเส้นทางที่เดินทางผ่านคือ S-E-F-G-T ตามรูปที่ 4-12

4.6.3 การพิสูจน์เส้นทางที่ดีที่สุดของ A*

การพิสูจน์ว่าเส้นทางที่เอสตาร์เลือกนั้นเป็นเส้นทางที่ดีที่สุด ตามรูปที่ 4-13 เป็นตัวอย่าง ของการเดินทางจากตำแหน่งเริ่มต้น ไปยังจุดหมาย G ที่เป็นเส้นทางที่ดีที่สุดโดยที่ n คือ โหนดปัจจุบันใด ๆ และ G_2 เป็นเส้นทางที่ดีที่สุดในลำดับรอง (suboptimal goal) ต่อไปจะเป็นการพิสูจน์ว่าเส้นทางที่เลือก G คือเส้นทางที่ดีที่สุด กำหนดให้ G คือโหนดจุดหมายที่ได้เส้นทางที่ดีที่สุด โดยมีค่าของฮิวริสติกฟังก์ชั่น คือ f*

รูปที่ 4-13 เปรียบเทียบการเดินทางที่ดีที่สุด

กำหนดให้ G_2 คือโหนดที่ได้เส้นทางที่ดีที่สุดในลำดับรอง ดังนั้นที่โหนดจุดหมายค่าของระยะทาง G_2 จะต้องมากว่า f* ซึ่งเขียนเป็นสมการได้

$$g(G_2) > f^*$$

การที่ A* เลือกเส้นทางที่ดีที่สุดนั้น ถ้ากำหนดให้ G_2 เป็นตำแหน่งสุดท้ายแล้วต้องการให้เป็นเส้นทางที่ดีที่สุดรอง ดังนั้นในการเลือกเส้นทางที่ผ่าน n ที่ไปถึง G ตามรูปที่ 5.8 จะได้ค่าตาม (5.2)

$$f^* \geq f(n)$$

ดังนั้นถ้า n ไม่ได้ถูกเลือกโดย G_2 ค่าของ f(n) จะต้องมากกว่า $f(G_2)$ ด้วยตาม (5.3)

$$f(n) \geq f(G_2)$$

เมื่อการรวมกันของทั้งสอง (5.2) กับ (5.3) จะได้

$$f^* \geq f(G_2)$$

ดังนั้นถ้ากำหนดให้ตำแหน่งจุดหมายมีค่า $h(G_2) = 0$ จะได้ว่า $f(G_2) = g(G_2)$

$$f^* \geq g(G_2)$$

ดังนั้นจึงสามารถพิสูจน์ได้ว่าเส้นทางที่จุดหมายเป็น G จึงเป็นเส้นทางที่ดีที่สุด และ G_2 เป็นเส้นทางที่ดีที่สุดในลำดับรอง สำหรับรายละเอียดของการค้นหาจะได้อธิบายในขั้นตอนการค้นหาเส้นทางบนแผนที่

4.7 การค้นหาเส้นทางบนแผนที่

จากที่ได้อธิบายหลักการค้นหาแบบต่าง ๆ มาแล้ว เราจะนำหลักการค้นหาดังกล่าวมาใช้ในการค้นหาเส้นทางบนแผนที่ ซึ่งจะเป็นการค้นหาโหนดที่เป็นจุดเชื่อมของเส้นถนน โดยที่เมื่อนำโหนดทั้งหมดกับเส้นเชื่อมโหนดซึ่งก็คือ path มาต่อกันก็จะได้เป็นเส้นทางที่จะนำไปสู่จุดหมาย จะเป็นผลให้ทุก ๆ โหนดมีการเชื่อมต่อกัน สามารถเดินทางไปจากจุดหนึ่งไปยังอีกจุดหนึ่งได้ โดยทั่วไปแล้วสิ่งที่เราสนใจก็คือการเดินทางไปในกราฟนั้นเส้นทางใดที่ดีที่สุด ได้แก่ มีระยะทางสั้นที่สุด ใช้เวลาน้อยที่สุด เป็นต้น

วิธีการค้นหาที่ใช้งานได้ดีจะต้องมีการนำเอาข้อมูลมาช่วยในการตัดสินใจเลือกเส้นทาง ที่จะนำไปสู่จุดหมายได้โดยประหยัดทั้งเวลาและหน่วยความจำ ดังนั้นวิธีการค้นหาแบบฮิวริสติกแบบเอสตาร์ จึงถือว่าเป็นวิธีที่เหมาะที่สุดสำหรับปัญหาของเราเพราะสามารถรับประกันว่าให้คำตอบที่ดีที่สุด

4.8 การค้นหาเส้นทางที่ดีที่สุดโดยประยุกต์ใช้วิธีเอสตาร์

ในวิทยานิพนธ์นี้เราได้ประยุกต์ใช้การค้นหาแบบเอสตาร์ ซึ่งใช้ f(n) = g(n) + h(n) ในการชี้นำการค้นหาให้ไปสู่จุดหมายอย่างรวดเร็ว

สำหรับในระบบของเราจะให้ g(n) เป็นการคำนวณค่าระยะทางจริงจากจุด 2 จุด ระหว่าง path ซึ่งจะเห็นว่าสอดคล้องกับเงื่อนไขข้างต้น และให้ h(n) เป็นการคำนวณระยะทางจากจุดปลาย (โหนด n) ของเส้นทางที่อยู่ในระหว่างการค้นหาให้เป็นพิกัด (x_1, y_1) ไปยังจุดหมาย (โหนด g) ซึ่งกำหนดเป็นพิกัด (x_2, y_2) ดังนี้

$$h(n) = \sqrt{(y_2 - y_1)^2 + (x_2 - x_1)^2}$$

จะพบว่า h(n) เป็นการประมาณการระยะทางของเส้นทางจากโหนด n ไปยังโหนด g ซึ่งไม่มากกว่าระยะทางจริงของเส้นทางที่สั้นที่สุดที่เชื่อมจาก n ไปยัง g เสมอ เนื่องจากเส้นตรงที่เชื่อมระหว่าง n และ g ย่อมสั้นที่สุด (ดูจากรูป 4-14(ก)) ให้เส้นตรงเชื่อม n และ g เป็น *l* ส่วนเส้นทางในความเป็นจริงอาจจะมีหลายๆ path ที่เชื่อมจากโหนด n ไปยังโหนด g ซึ่ง path เหล่านี้อาจจะผ่านจุด x หรือทับเส้นตรง*l* ก็ได้ กรณีเส้นทางที่ทับ*l* ก็จะเป็นเส้นที่สั้นที่สุด ดังรูปที่ 4-14(ก)

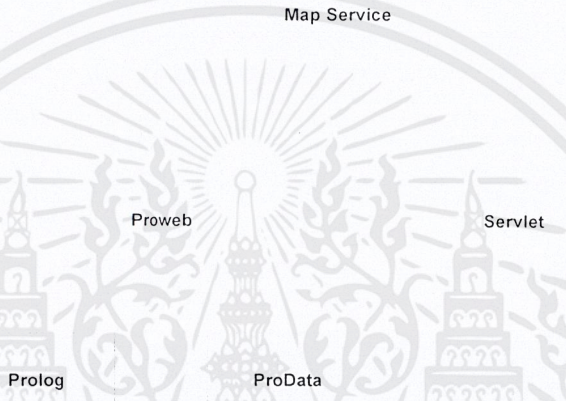
รูปที่ 4-14 ลักษณะของ function ต่าง ๆ (ก) function ของ h(n) (ข) function ของ f(n)

ดังนั้นการใช้ $f(n) = g(n) + h(n)$ ในที่นี้เราใช้กับการค้นหาเส้นทางการเดินทางที่สั้นที่สุด

บทที่ 5
การออกแบบและการสร้างเว็บไซต์

5.1 การออกแบบ

เราสามารถออกแบบเว็บไซต์ได้ดังไดอะแกรมในรูปที่ 5-1



รูปที่ 5-1 ไดอะแกรมของเว็บไซต์

เว็บแผนที่นี้จะแบ่งออกเป็น 2 ส่วนใหญ่ๆ คือ

1. ส่วนติดต่อกับผู้ใช้(User Interface) จะใช้โปรเว็บในการสร้างเว็บโดยจะแบ่งออกเป็น 2 ส่วนคือส่วนวินิจฉัยจะใช้โปรล็อกและฐานข้อมูลจะใช้โปรดาต้า
2. ส่วนแสดงผลภาพแผนที่ จะใช้ Servlet ในการนำภาพแผนที่ที่ได้มาแสดงผลบนเว็บ

5.2 การสร้างเว็บไซต์

การสร้างเว็บเราได้แยกออกเป็นส่วนได้ดังนี้

5.2.1 โปรเว็บ

จะทำหน้าที่รับข้องมูลเข้ามาเพื่อส่งไปยังส่วนวินิจฉัยแล้วนำผลที่ได้นำมาแสดงผลในรูปของเว็บเพ็จ การแสดงผลรวมไปถึงภาพแผนที่ที่ได้จากผลการวินิจฉัยและการสร้างจาวาสคริปต์ด้วย แบ่งการแสดงออกตามการทำงานของเว็บไซต์ได้ดังนี้

1. การค้นหาสถานที่ จะเป็นแสดงผลรายชื่อสถานที่และประเภทที่ต้องการค้นหา ไคเอนท์สามารถเลือกได้เพียงสถานที่เพียงแห่งเดียวเท่านั้นในการแสดงผลโดยใช้แท็ก INPUT TYPE= RADIO เช่น

<input type=radio name="road_id" value="road_lad">ลาดพร้าว

2. การค้นหาสถานที่ใกล้เคียง เมื่อผู้ใช้ข้อมูลประเภทสถานที่และรัศมีการค้นหาแล้ว โปรเว็บก็จะแสดงผลเพื่อให้ผู้ใช้ป้อนข้อมูลของจุดอ้างอิงหรือเลือกที่ตัวแผนที่ได้เลยก็สามารถระบุจุดอ้างอิงได้ด้วย หลังจากนั้นเมื่อได้รับข้อมูลทั้งหมดมาก็นำไปประมวลผลในส่วนวินิจฉัยแล้วแสดงผลรายชื่อสถานที่ค้นหาได้และแสดงตำแหน่งของจุดอ้างอิงและสถานที่ต่างๆในแผนที่ด้วย
3. การค้นหาถนนสายหลัก จะหลักการทำงานคล้ายกับการค้นหาสถานที่แต่จะต่างกันที่ส่วนแสดงผลลัพธ์โดยจะแสดงเส้นทางของถนนนั้นๆและชื่อถนนด้วย
4. การค้นหาเส้นทางที่สั้นที่สุด จะแสดงผลเพื่อรับตำแหน่งของจุดเริ่มต้นและจุดหมาย และแสดงผลการวินิจฉัยออกเป็นเส้นทางที่ได้ในแผนที่และแสดงรายละเอียดของเส้นทางด้วย

5.2.2 ส่วนวินิจฉัย

แบ่งออกตามหน้าที่การทำงานของเว็บไซต์

1. การค้นหาสถานที่ จะเป็นการค้นหาสถานที่และประเภทที่กำหนดโดยจะเก็บเป็นลิสต์ ส่วนการนำไปใช้งานอาจจะมีการดัดแปลงเพื่อความเหมาะสมกับงาน

```
search_PlaceList( PlaceType, Name , List ):-
    findall( PlaceID ,
            (
            place( PlaceID, _, _, Type, PlaceName),
            sub_string( Name, PlaceName ),
            ),
            List).
```

2. การค้นหาสถานที่ใกล้เคียง จะเป็นการค้นหาสถานที่ที่เป็นประเภทที่กำหนดและอยู่ในระยะที่กำหนดโดยจะเก็บเป็นลิสต์

```
nearest_place( X1, Y1, Range, PlaceType, PlaceList ):-
    findall( [PlaceID,PlaceName],
                    (
                    place( PlaceID, X2, Y2, PlaceType, PlaceName ),
                     dist_PointToPoint(  X1,Y1, X2, Y2, Dist ),
                     Dist < Range
                    ),
                    PlaceList ).
```

3. การค้นหาถนนสายหลัก จะหลักการทำงานคล้ายกับการค้นหาสถานที่

```
search_RoadList( Sub, List ):-
    findall( RoadID ,
            (
```

```
        road( RoadID, RoadName, _, _, _ ),
        sub_string( Sub, RoadName
        ),
        List ).
```

4. การค้นหาเส้นทางที่สั้นที่สุด จะเป็นการค้นหาเส้นทางที่สั้นที่สุดโดยวิธีเอ-สตาร์ซึ่งมีรายละเอียดอยู่ในหัวข้อ 4.8 โดยนำหลักการดังกล่าวมาเขียนในภาษาโปรล็อกดั้งนี้

```
searchNode( PointID, PointID, [] , 0 ).
searchNode( StartPointID, GoalPointID, Solution, Distance ):-
    astar_node( [ [ 0, 0, StartPointID ] ], Solution, GoalPointID, Distance ),
    !.
astar_node( [ [ _, Distance, Goal|Path ] | _ ], [ Goal|Path ].  Goal, Distance ).
astar_node( [ Path|Paths ], Solution, GoalNode, Distance ) :-
    extendNode( Path, NewPaths, GoalNode ),
    insertAll( NewPaths, Paths, Paths2 ),
    astar_node( Paths2, Solution, GoalNode, Distance ).

extendNode( [ F, G, ThisPointID|Path ], NewPaths, GoalPoinID ) :-
    setof(  [Fn, Gn, NewPointID, ThisPointID|Path],
            (
            linked( ThisPointID, NewPointID ),
            \+ member( NewPointID, [ ThisPointID|Path ] ),
            dist_PtIDToPtID( ThisPointID, NewPointID, Dist ),
            Gn is G+Dist,
            heuristicFn( NewPointID, GoalPoinID, H ), % Heuristic Funtion
            Fn is Gn+H
            ),
            NewPaths ).
linked( A, B ):-
    ( path( _, A, B );path( _, B, A ) ).
dist_PtIDToPtID( PtId1, PtId2, Dist ):-
    position( PtId1,  X1, Y1  ),
    position( PtId2,  X2, Y2  ),
    dist_PointToPoint(  X1,Y1, X2, Y2, Dist ).
heuristicFn( PtId1, PtId2, Dist ):-
    dist_PtIDToPtID( PtId1, PtId2, Dist ).
```

```
insertAll( [ ], Paths, Paths ).
insertAll( [ Path|Paths ], Paths2, Paths3 ) :-
    insertOne( Path, Paths2 , Paths4 ),
    insertAll(Paths,Paths4,Paths3).
insertOne( Path, [ ], [ Path ] ).
insertOne( [ F, G|Path ], [ [ F2, G2|Path2 ]|Paths ], [ [ F, G|Path ], [ F2, G2|Path2 ]|Paths ] ):-
    F < F2.
insertOne( [ F, G|Path ], [ [ F2, G2|Path2 ]|Paths ], [ [ F2, G2|Path2 ]|Paths2 ] ):-
    insertOne( [ F, G|Path ], Paths, Paths2).
```

5.2.3 ฐานข้อมูล

จะเก็บข้อมูลเฉพาะฐานข้อมูลเท่านั้นหมายถึง fact เท่านั้นที่สามารถเก็บไว้ในฐานข้อมูลได้จะมีด้วยกันทั้งหมด 8 ตารางดังนี้

1. ตาราง POSITION เป็นตารางที่เก็บตำแหน่งของจุด(Node)ต่างๆ

ชื่อคอลัมน์	รายละเอียด
POINTID	รหัสจุด
X	ลองติจูด
Y	ละติจูด

ตารางที่ 5-1 ตาราง POSITION

2. ตาราง PATH เป็นตารางที่เก็บชื่อพาทและของจุด 2 จุดต่างๆ

ชื่อคอลัมน์	รายละเอียด
PATHID	รหัสพาท
POINTID1	รหัสจุด
POINTID1	รหัสจุด

ตารางที่ 5-2 ตาราง PATH

3. ตาราง PLACE เป็นตารางที่เก็บรายละเอียดสถานที่

ชื่อคอลัมน์	รายละเอียด
PLACEID	รหัสสถานที่
X	ลองติจูดที่แสดงชื่อถนน
Y	ละติจูดที่แสดงชื่อถนน
PLACETYPE	ประเภทของสถานที่
PLACENAME	ชื่อสถานที่

ตารางที่ 5-3 ตาราง PLACE

4. ตาราง PLACETYPE เป็นตารางที่เก็บประเภทสถานที่

ชื่อคอลัมน์	รายละเอียด
PLACETYPE	รหัสประเภทสถานที่
PLACETYPENAME	ประเภทของสถานที่

ตารางที่ 5-4 ตาราง PLACETYPE

5. ตาราง ROAD เป็นตารางที่เก็บรายละเอียดถนน

ชื่อคอลัมน์	รายละเอียด
ROADID	รหัสถนน
ROADNAME	ชื่อถนน
ROADTYPE	รหัสประเภทถนน
X	ลองติจูด
Y	ละติจูด

ตารางที่ 5-5 ตาราง ROAD

6. ตาราง ROADPATH เป็นตารางที่เก็บบอกได้ว่าถนนมีจุดอะไรบ้าง

ชื่อคอลัมน์	รายละเอียด
ROADID	รหัสถนน
POINTID	รหัสจุด
TYPE	ประเภทของจุด

ตารางที่ 5-6 ตาราง ROADPATH

7. ตาราง ROADTYPE เป็นตารางที่เก็บชื่อพาทและของจุด 2 จุดต่างๆ

ชื่อคอลัมน์	รายละเอียด
ROADTYPE	รหัสประเภทถนน
ROADTYPENAME	ชื่อประเภทถนน

ตารางที่ 5-7 ตาราง ROADTYPE

8. ตาราง CROSSROAD เป็นตารางที่เก็บรายละเอียดทางแยกต่างๆ

ชื่อคอลัมน์	รายละเอียด
CROSSID	รหัสทางแยก
POINTID	รหัสจุด

ตารางที่ 5-8 ตาราง CROSSROAD

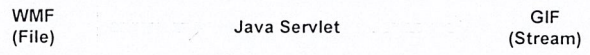
5.2.4 Servlet

5.2.4.1 เหตุผลการใช้งาน Servlet

เนื่องมาจาก WIN-PROLOG สามารถสร้างไฟล์ภาพเป็นแบบเวคเตอร์แต่ไม่สามารถแสดงผลในเว็บบราวเซอร์ได้ จึงต้องมีโปรแกรมหนึ่งที่เป็นตัวกลางในการแปลงไฟล์ดังกล่าวให้เป็นไฟล์ประเภทอื่นเช่น GIF, JPEG เป็นต้น แต่เราเลือกที่จะแปลงเป็นไฟล์ GIF เนื่องจากหาโค้ดแปลงเป็นไฟล์ GIF ที่เป็น Open source ได้ โปรแกรมที่เรานำมาเป็นตัวกลางในการแปลงเราเลือกใช้ JavaServlet เพราะสามารถแปลงแล้วส่งเป็นข้อมูลภาพโดยที่ไม่ต้องเก็บเป็นๆไฟล์ก่อนหรือเรียกว่า แปลงแบบไดนามิก

5.2.4.2 การทำงาน

ลักษณะการทำงานของ Servlet เป็นดังไดอะแกรมในรูปที่ 5-2

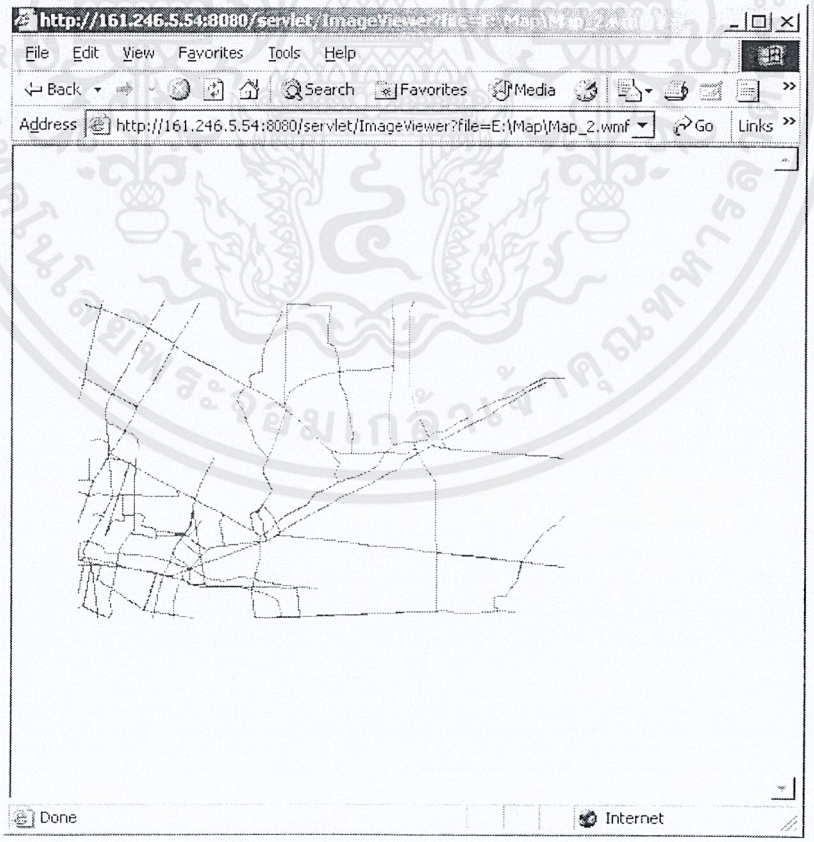


รูปที่ 5-2 ไดอะแกรมการทำงานของ Servlet

เนื่องจากไฟล์ WMF เป็นไฟล์แบบเวคเตอร์ดังนั้นเมื่อมีการแปลงไฟล์ Servlet จะทำการอ่านไฟล์ WMF เพื่อจะวาดภาพหลังจากนั้นก็จะแปลง ภาพที่ได้เป็นข้อมูลในรูปแบบของไฟล์ GIF ดังในรูปที่ 5-3

5.2.4.3 ปัญหาในการสร้าง

การหาข้อมูลรูปแบบของไฟล์ WMF ในการทำนั้นยากมากเนื่องจากเป็นรูปแบบที่ไมโครซอฟท์ค้นคิดขึ้นมาแต่ไม่มีข้อมูลของไฟล์ดังกล่าว ดังนั้นจึงต้องค้นหาตัวอย่างโปรแกรมที่สามารถแสดงผลไฟล์ WMF ได้ ซึ่งเป็นโปรแกรม WmfView [8] แล้วต้องนำโปรแกรมดังกล่าวมาดัดแปลงให้สามารถทำงานได้



รูปที่ 5-2 ตัวอย่างการแปลงไฟล์ WMF ผ่าน Servlet

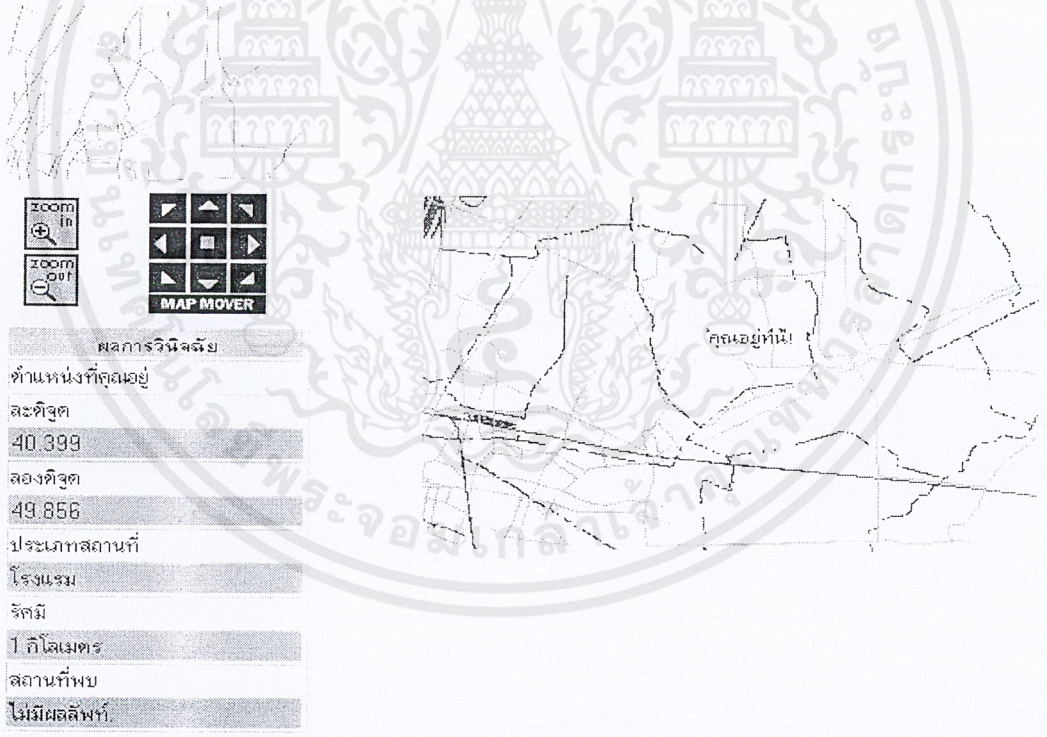
บทที่ 6
ผลการทดลอง

ระบบให้บริการแผนที่บนเว็บนี้พัฒนาขึ้นโดยใช้ภาษา PROLOG ซึ่ง Software ที่ใช้คือ WIN-PROLOG 4.040 และโปรเว็บซึ่งเป็นภาษาโปรล็อกสำหรับใช้พัฒนาโปรแกรมประยุกต์บนเครื่องเว็บเซิร์ฟเวอร์ สำหรับส่วนฐานข้อมูลของแผนที่นั้นพัฒนาโดยอาศัยโปรดาต้า

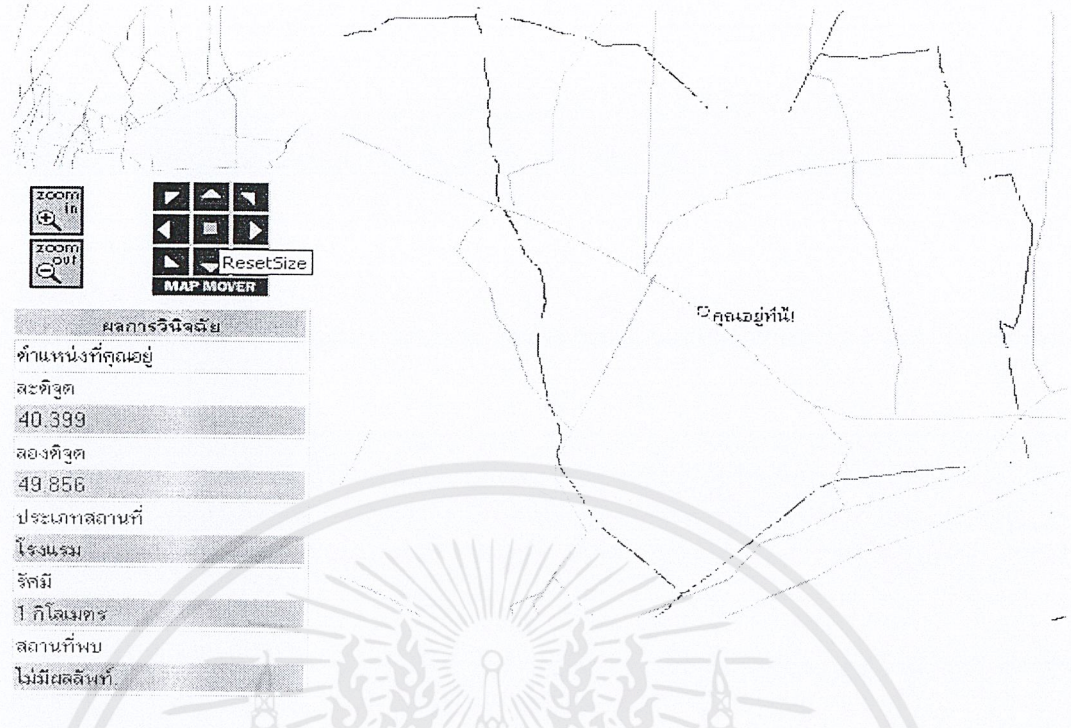
สำหรับในการทดลองนี้แบ่งการทดลองออกเป็น 6 ส่วนคือ การทดลองย่อ-ขยาย และเลื่อนแผนที่ การค้นหาถนนสายหลัก การค้นหาสถานที่สำคัญ การค้นหาสถานที่ใกล้เคียง การค้นหาเส้นทางที่สั้นที่สุด และวิจารณ์ผลการทดลอง

6.1 การทดลองย่อ-ขยาย และเลื่อนแผนที่

ในส่วนของแผนที่นี้เราสามารถที่จะย่อหรือขยายแผนที่ เพื่อดูรายละเอียดในจุดต่างๆได้ โดยการกดปุ่ม Zoom in และ Zoom Out บนแผนที่ ดังแสดงในรูปที่ 6-1, 6-2 และรูปที่ 6-3 แสดงการเลื่อนแผนที่ โดยสามารถเลื่อนได้ทั้ง 8 ทิศทาง



รูปที่ 6-1 แสดงแผนที่ขนาดย่อ



รูปที่ 6-2 แสดงการขยายแผนที่

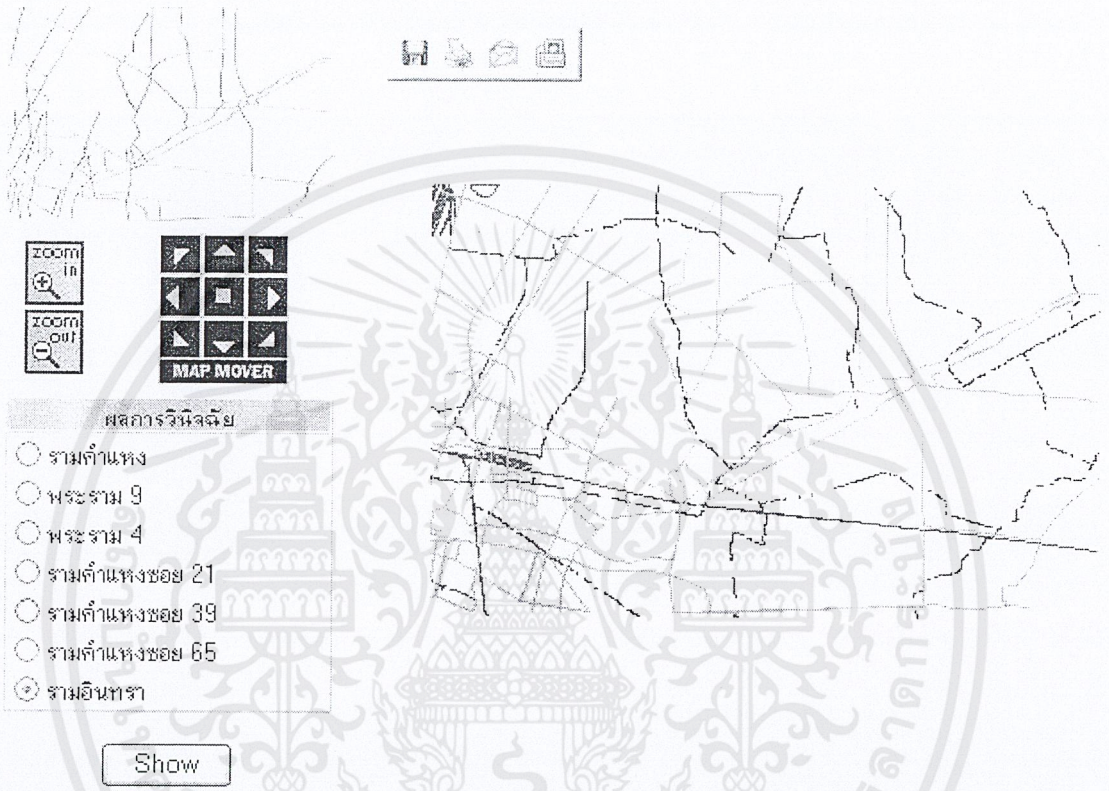
รูปที่ 6-3 แสดงการเลื่อนแผนที่

6.2 การทดลองค้นหาถนนสายหลัก

วิธีการทดลองคือ

1. ให้ใส่ชื่อของถนนที่ต้องการจะค้นหา เช่นต้องการ ค้นหาถนนรามอินทรา ให้ใส่คำว่า "ราม" หรือ "อินทรา"

2. กดปุ่ม"ค้นหา"จากนั้นรอสักครู่ โปรแกรมจะแสดงชื่อของถนนที่มีคำว่า"ราม"หรือ"อินทรา" ขึ้นมา ดังรูปที่ 6-4



รูปที่ 6-4 แสดงการค้นหาถนนสายหลัก

3. ให้เลือกชื่อถนนที่ต้องการจะค้นหาจากรายการที่แสดงขึ้นมา

4. กดปุ่ม "Show" โปรแกรมก็จะแสดงถนนที่ค้นหาบนแผนที่ ดังรูปที่ 6-5

รูปที่ 6-5 แสดงการค้นหาถนนสายหลัก

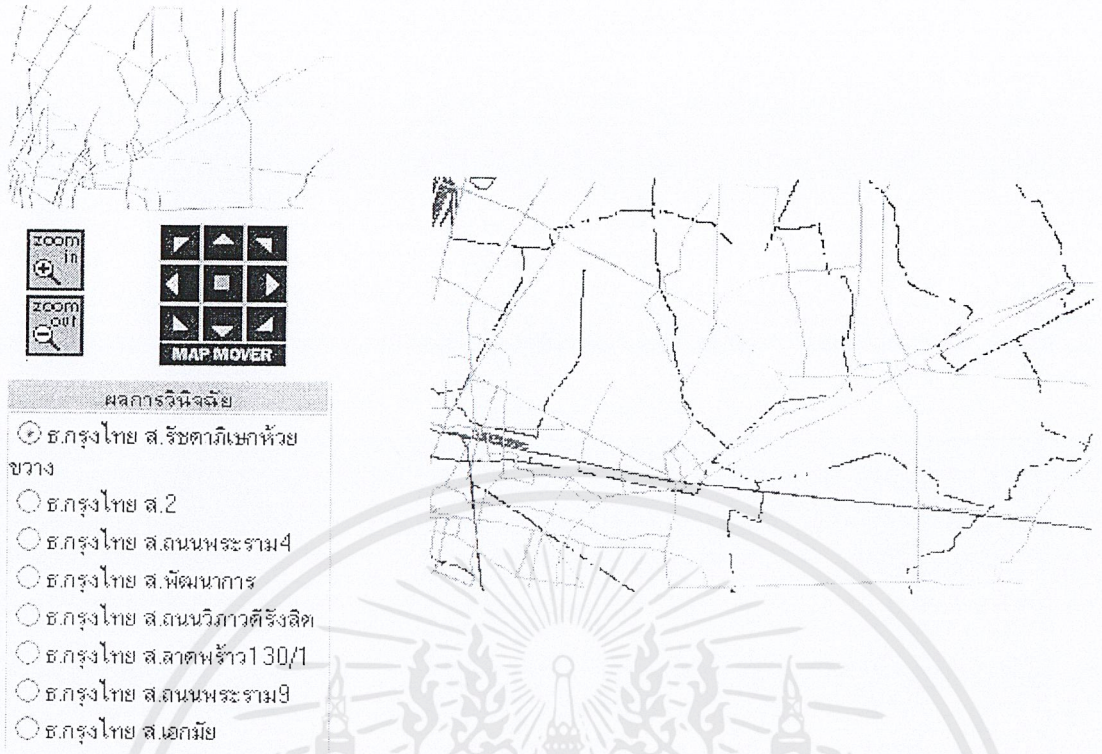
6.3 การทดลองค้นหาสถานที่สำคัญ

วิธีการทดลองคือ

1.เลือกประเภทของสถานที่ที่ต้องการจะค้นหา

2.ให้ใส่ชื่อของสถานที่ที่ต้องการค้นหา เช่น ต้องการหาธนาคารกรุงไทย ก็ให้ใส่คำว่า"กรุง"หรือคำว่า"ไทย"

3.กดปุ่ม "ค้นหา" จากนั้นรอสักครู่ โปรแกรมจะแสดงชื่อของธนาคารที่มีคำว่า"กรุง"หรือ"ไทย"ขึ้นดังรูปที่ 6-6



รูปที่ 6-6 แสดงการค้นหาสถานที่สำคัญ

4. ให้เลือกสถานที่ ที่ต้องการจะค้นหาจากรายการที่แสดงขึ้นมา
5. กดปุ่ม "Show" โปรแกรมจะแสดงสถานที่ที่ค้นหาบนแผนที่ ดังรูปที่ 6-7

รูปที่ 6-7 แสดงการค้นหาสถานที่สำคัญ

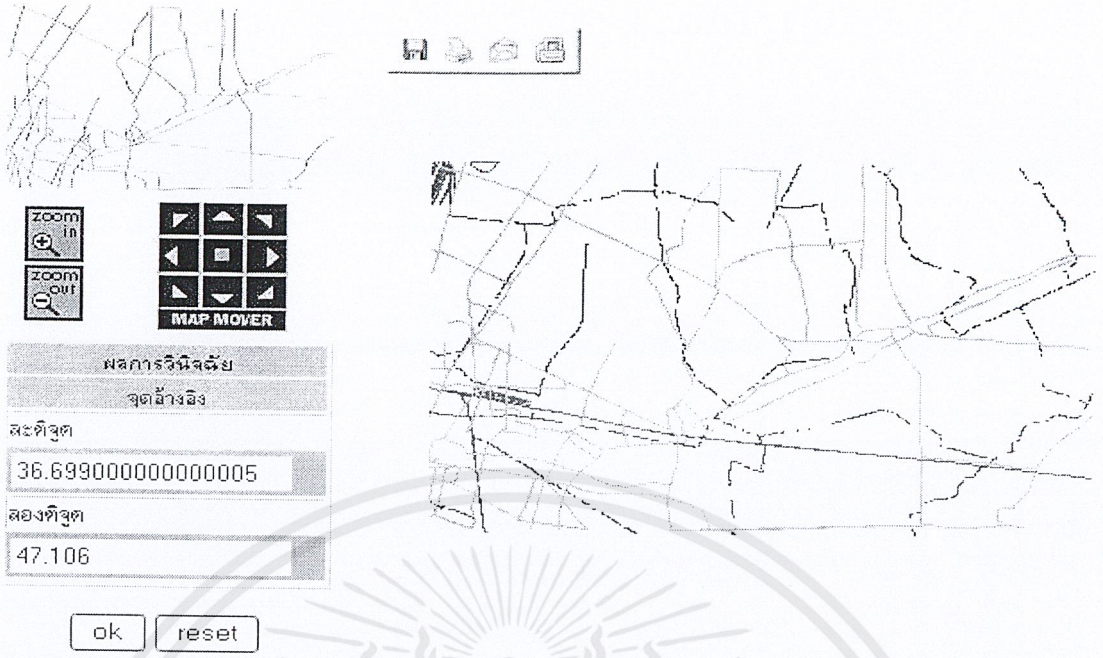
ถ้าต้องการดูในอัตราส่วนที่เล็กลง ก็ให้กดปุ่มซูมเอาท์บนแผนที่ จะได้ผลลัพท์ดังรูปที่ 6-8

รูปที่ 6-8 แสดงการค้นหาสถานที่สำคัญ

6.4 การทดลองค้นหาสถานที่ใกล้เคียง

วิธีการทดลองคือ

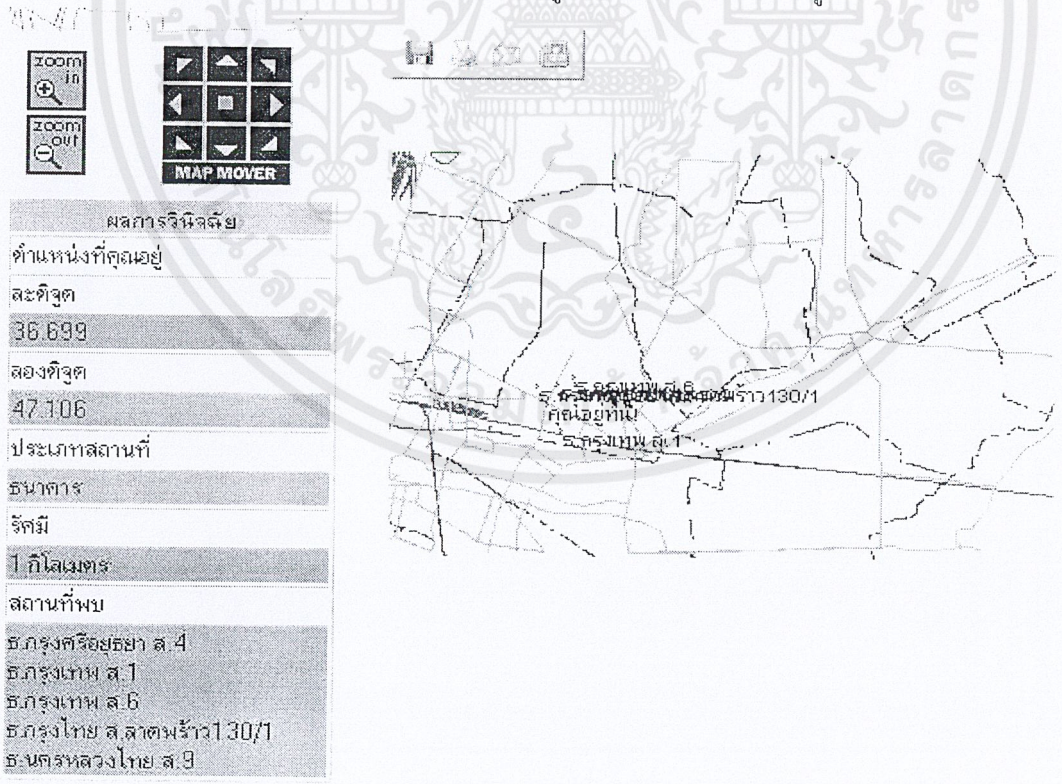
1. ให้เลือกประเภทของสถานที่ ที่ต้องการค้นหา เช่น ธนาคาร หรือ โรงแรม เป็นต้น
2. ให้เลือกรัศมีที่ต้องการค้นหา
3. กดปุ่ม "ค้นหา" โปรแกรม ก็จะแสดงแผนที่ขึ้นมา ดังรูปที่ 6-9



รูปที่ 6-9 แสดงการค้นหาสถานที่ใกล้เคียง

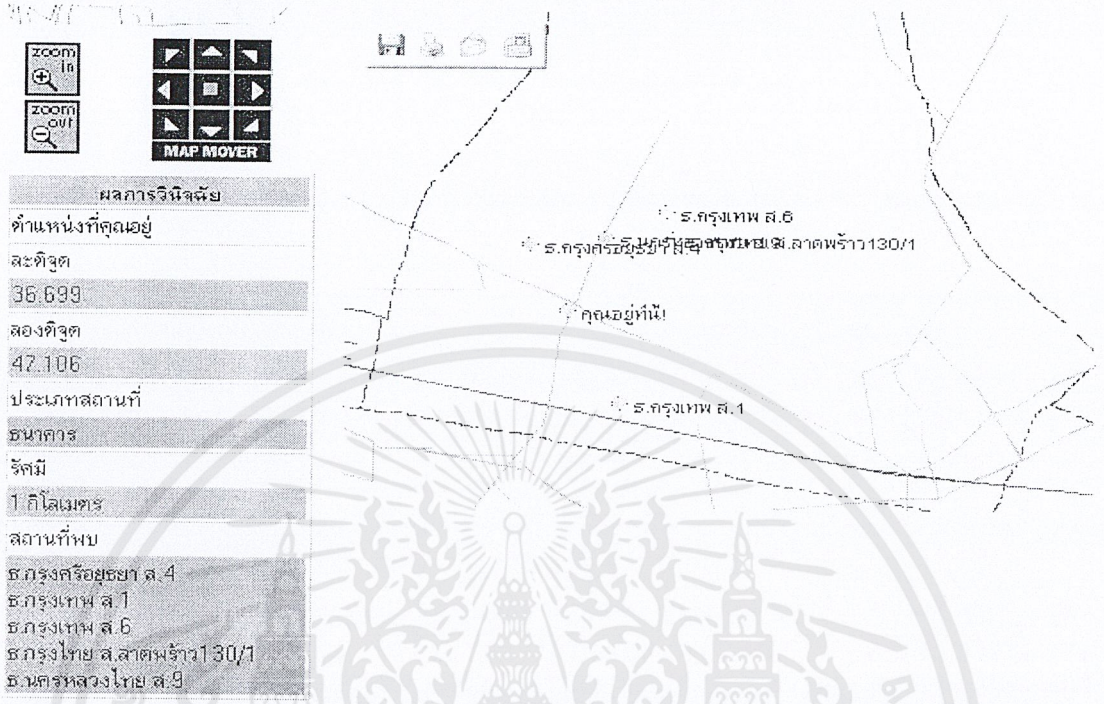
4. ให้ click เมาส์บนแผนที่ ตรงตำแหน่งที่ต้องการใช้เป็นศูนย์กลาง

5. กดปุ่ม"Search"โปรแกรมก็จะแสดงสถานที่ที่อยู่ในรัศมีที่เลือกขึ้นมา ดังรูปที่ 6-10



รูปที่ 6-10 แสดงการค้นหาสถานที่ใกล้เคียง

สถานที่ที่แสดงจะมีขนาดเล็ก ไม่สามารถมองเห็นรายละเอียดได้ ถ้าต้องการดูรายละเอียดต่างๆเกี่ยวกับแผนที่จะต้องซูมเข้าไป ดังรูปที่ 6-11 จะสามารถเห็นรายละเอียดต่างๆได้ชัดเจนขึ้น

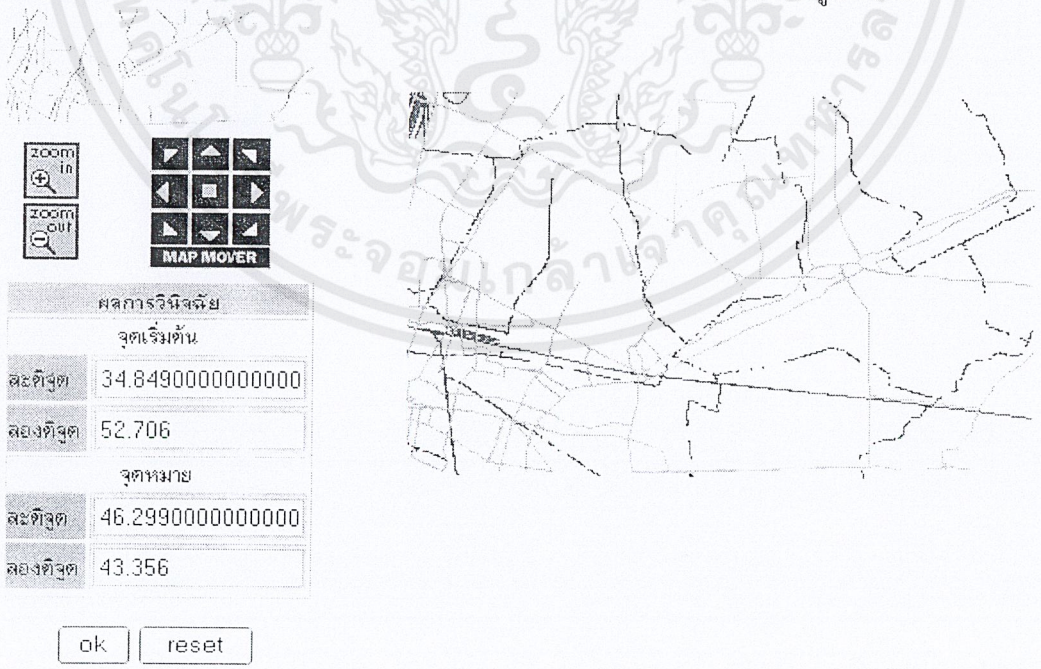


รูปที่ 6-11 แสดงการค้นหาสถานที่ใกล้เคียง

6.5 การทดลองค้นหาเส้นทางที่สั้นที่สุด

วิธีการทดลองคือ

1. กดปุ่ม “ตกลง” เพื่อเริ่มต้นทำการค้นหาเส้นทาง โดยจะมีแผนที่แสดงขึ้นมา ดังรูปที่ 6-12



รูปที่ 6-12 แสดงการค้นหาเส้นทางที่สั้นที่สุด

2. เลือกตำแหน่งที่จะใช้เป็นจุดเริ่มต้นโดยการ click บนแผนที่

3. เลือกตำแหน่งที่เป็นจุดหมาย

4. กดปุ่ม "Search" แล้วรอสักครู่ โปรแกรมจะแสดงเส้นทางขึ้นมาโดยใช้เส้นสีน้ำเงินเป็นเส้นทางที่สั้นที่สุด ดังรูปที่ 6-13

รูปที่ 6-13 แสดงการค้นหาเส้นทางที่สั้นที่สุด

6.6 วิจารณ์ผลการทดลอง

1. ในส่วนของแผนที่ชาญฉลาด ในส่วนของฐานความรู้ที่ใช้หลักการแทนความรู้ จะมีข้อจำกัดอยู่ที่ปริมาณข้อมูล ถ้าข้อมูลมากการประมวลผลของคอมพิวเตอร์อาจจะใช้เวลามากขึ้น ทั้งนี้เนื่องจากข้อมูลอยู่ในรูปของฐานความรู้ที่อยู่ในรูปของเพรดดิเคททางตรรกศาสตร์ เช่น การค้นหาเส้นทางที่สั้นที่สุด อาจจะต้องใช้เวลานานถ้ามีข้อมูลมาก (โหนดมาก)

2. สำหรับในการแทนความรู้ที่เกี่ยวกับถนนนั้นในบางครั้งอาจจะคลาดเคลื่อนกับข้อมูลจริงบ้าง ทั้งนี้เพราะถนนบางแห่งอาจจะมีการตัดถนนหรือขยายถนน เป็นต้น

3. ในส่วนของข้อมูลสถานที่อาจจะมีความผิดพลาดบ้าง เนื่องจากฐานความรู้ที่ใช้เป็นฐานความรู้เก่าข้อมูลอาจจะไม่ตรงกับในปัจจุบันหรือ บางครั้งข้อมูลอาจจะผิดพลาดเนื่องจากป้อนข้อมูลเข้าไปผิด ทำให้การค้นหาคลาดเคลื่อนไปด้วย

บทที่ 7
บทวิจารณ์และสรุป

7.1 บทวิจารณ์และสรุป

ระบบให้บริการแผนที่บนเว็บที่พัฒนาขึ้นมานี้ทำงานบนโปรเว็บเซิร์ฟเวอร์สามารถทำการวินิจฉัยฐานความรู้แผนที่ได้อย่างดี โดยมีให้บริการได้ดังนี้ สอบถามตำแหน่งของสถานที่สำคัญได้ สอบถามหาสถานที่ใกล้เคียง สอบถามตำแหน่งของถนน และสอบถามเส้นทางที่สั้นที่สุด ซึ่งผลลัพธ์ที่ได้จะส่งกลับไปให้ผู้ใช้เป็นภาพแผนที่

เมื่อเปรียบเทียบกับเว็บไซต์ที่ให้บริการแผนที่ในปัจจุบัน เว็บไซต์ที่พัฒนาขึ้นมามีความสามารถมากกว่าคือ สามารถสอบถามเส้นทางที่สั้นที่สุดได้ แต่ก็มีข้อด้อยกว่าคือยังมีฐานความรู้แผนที่น้อยไป

7.2 ปัญหาที่เกิดขึ้น

1. เอกสารที่เกี่ยวกับไฟล์ WMF ไม่มีทำให้ยากต่อการศึกษา และสร้างโปรแกรมแปลงไฟล์ WMF เป็นก็ไปได้อย่างล่าช้า
2. ข้อจำกัดของโปรเว็บที่ไม่สามารถดีบักได้ จึงทำให้ยากต่อการพัฒนาโปรแกรม
3. โปรเว็บในเวอร์ชั่นที่ใช้อยู่มีประสิทธิภาพน้อยคือ ต้องใช้เวลาในการประมวลผลนาน เนื่องจากจะประมวลผลทุกครั้งที่มีการติดต่อไปยังโปรแกรมบนโปรเว็บแม้จะเป็นการสนทนาเดิมที่ต่อเนื่องกันก็ตาม
4. ข้อมูลที่ได้จากโปรแกรมระบบนำร่องชาญฉลาด[1] ทำความเข้าใจได้ยาก ทำให้ไม่ทราบสาเหตุของการทำงานที่ผิดพลาดของโปรแกรมได้

7.3 วิธีการแก้ปัญหา

ปัญหาส่วนใหญ่ที่เกิดขึ้นส่วนใหญ่ต้องใช้เวลามากในการแก้ปัญหา เนื่องจากไม่มีข้อมูลและข้อจำกัดของตัวโปรเว็บ ทำให้ต้องใช้เวลาในการศึกษามาเป็นพิเศษ วิธีแก้ปัญหาก็คือ ค้นหาข้อมูลในอินเตอร์เน็ทและสอบถามไปยังบริษัท LPA ส่วนของโปรเว็บถ้าต้องการนำไปใช้งานจริงต้องใช้เวอร์ชั่นใหม่ที่มีประสิทธิภาพมากขึ้น โดยสามารถเก็บผลของการทำงานในครั้งก่อนในการสนทนาครั้งเดียวกันได้ หรือใช้เซิร์ฟเวอร์ที่มีความเร็วสูง ในส่วนข้อมูลที่ได้จากโปรแกรมระบบนำร่องชาญฉลาดสามารถทำความเข้าใจได้แต่ต้องใช้เวลาในการทำความเข้าใจ

7.4 แนวทางการพัฒนา

1. สามารถเพิ่มข้อมูลต่างของแผนที่เพื่อให้ครอบคลุมพื้นที่ทั้งหมดของกรุงเทพ
2. สามารถพัฒนาซอร์สโค้ดให้สามารถวินิจฉัยได้มากขึ้นเช่น ค้นหาเส้นทางที่ใช้เวลาเดินทางน้อยที่สุด ค้นหาเส้นทางที่เดินทางสะดวกที่สุด
3. สามารถปรังปรุงให้มีความฉลาดมากขึ้นได้โดยการปรับปรุงส่วนการวินิจฉัยฐานความรู้
4. สามารถเพิ่มข้อมูลที่เกี่ยวกับการเดินรถเมล์เพื่อทำให้สามารถบริการข้อมูลเกี่ยวกับรถเมล์ได้ เช่น สอบถามเส้นทางการเดินรถของรถเมล์ได้ สอบถามการเดินทางว่าต้องขึ้นรถเมล์สายอะไรบ้างถึงจะถึงจุดหมาย

ภาคผนวก ก.

Java Script & Servlet

1. จาวา (Java)

Java คือภาษาโปรแกรมคอมพิวเตอร์ภาษาหนึ่ง ที่ได้รับการพัฒนาโดยบริษัท Sun Microsystems กำเนิดของภาษาจาวามีสาเหตุเริ่มต้นจากความยุ่งยากในการพัฒนาโปรแกรมประยุกต์ (application) สำหรับระบบอินเทอร์เน็ต ทั้งนี้เพราะระบบอินเทอร์เน็ต เป็นระบบเปิดที่สามารถใช้งานจากเครื่องคอมพิวเตอร์ประเภทใด ๆ ก็ได้ ไม่ว่าจะเป็นเครื่องพีซี, แมคอินทอช, ซัน, เครื่องมินิคอมพิวเตอร์ ไปจนถึงเครื่องระดับซูเปอร์คอมพิวเตอร์ พ.ศ. 2534 บริษัท ซัน ไมโครซิสเต็มส์ (Sun Microsystems Inc.) จึงได้พัฒนาภาษาคอมพิวเตอร์ใหม่ที่มีประสิทธิภาพในการทำงานชนิดไม่ยึดติดกับแพลตฟอร์มขึ้นมา

จาวาจะทำการคอมไพล์คำสั่งจากซอร์สโค้ดให้กลายเป็นรหัสภาษากลางที่เรียกว่า ไบต์โค้ด (byte code) มีคุณลักษณะเด่นคือ มีขนาดเล็ก สามารถนำไปประยุกต์ใช้งานได้สะดวกรวดเร็ว โดยเตรียมโปรแกรมไว้บนเครื่องเซิร์ฟเวอร์และเมื่อมีการเรียกใช้งานจากเว็บบราวเซอร์ เซิร์ฟเวอร์จะทำการส่งข้อมูลโปรแกรมดังกล่าวกลับ เพื่อให้เว็บบราวเซอร์สั่งให้ทำงาน (run) ต่อไป

2. Java Script

Java Script นั้นถูกพัฒนาโดยทีมงาน Netscape เป็นโปรแกรมที่ทำงานเป็นสคริปต์เล็กๆ ใช้งานง่าย ทำงานโดยตัวโปรแกรมคำสั่ง อยู่ในซอร์สโค้ดของเว็บเพ็จ

Java Script ใช้ประโยชน์สำหรับงานด้านต่าง ๆ ทั้งการคำนวณการแสดงผล การรับ-ส่งข้อมูล และที่สำคัญคือสามารถโต้ตอบกับผู้ใช้ได้อย่างทันที ทั้งนี้จะช่วยให้เว็บเพ็จมีการเคลื่อนไหว เว็บเพ็จดูมีลูกเล่นมากขึ้น

การติดต่อสื่อสารระหว่างผู้ใช้กับเว็บเพ็จนั้น ต้องติดต่อโดยผู้ใช้ส่งข้อมูลผ่านทางแบบฟอร์มที่มีการเขียนโปรแกรม CGI รองรับไว้ในเครื่องเซิร์ฟเวอร์ เมื่อมีการส่งข้อมูลจากแบบฟอร์มมายังเครื่องเซิร์ฟเวอร์โปรแกรม CGI จะประมวลผลข้อมูลทุกครั้ง แล้วนำผลลัพธ์ที่ได้ส่งกลับคืนไปยังเครื่องคอมพิวเตอร์ของผู้ใช้ในรูปของเว็บเพ็จ HTML ใหม่ ลองนึกดูว่ากรณีที่ต้องมีการประมวลผลในลักษณะอย่างนี้ซ้ำ ๆ กันหลาย ๆ ครั้ง CPU ที่เครื่องเซิร์ฟเวอร์จะทำงานซ้ำ ๆ กันมากเกินความจำเป็นด้วยเหตุนี้จึงมีการนำเอา JavaScript เข้ามาแก้ไขจุดบกพร่องดังกล่าว โดยให้เครื่องคอมพิวเตอร์ของผู้ใช้ทำหน้าที่ประมวลผลเอง เพื่อช่วยลดภาระการทำงานของ CPU ที่เซิร์ฟเวอร์

ภาษา Java Script นั้น จะคล้ายกับ ภาษาจาวาแต่เข้าใจง่ายกว่าจาวา การเขียน Java Script ก็ใช้เพียง text editor ทั่วไป และ เว็บบราวเซอร์ที่สนับสนุนภาษา Java Script

โปรแกรม Microsoft Internet Explorer, Netscape Navigator และ Opera มีความ แตกต่างกันในการสนับสนุน Java Script เช่นตัว Object ที่อ้างอิงถึงเว็บบราวเวอร์ต่างกัน บางที function เหมือนกันแต่ให้ผลลัพธ์ต่างกัน ผู้เขียน Java Script ควรจะระมัดระวังในจุดนี้ และเขียนโปรแกรมให้สามารถใช้งานได้หลายบราวเซอร์เลยยิ่งดี

ถ้าจำแนก Java Script ออกตามลักษณะการทำงานทางฝั่ง ไคลเอนต์และฝั่งเซิร์ฟเวอร์ จะจำแนกออกได้เป็น 2 แบบดังนี้

1. Navigator JavaScript เป็น Client-Side JavaScript ซึ่งหมายถึง JavaScript ที่ถูกแปลทางฝั่งไคลเอนต์ จึงมีความเหมาะสมต่อการใช้งานของผู้ใช้ทั่วไปเป็นส่วนใหญ่
2. LiveWire JavaScript เป็น Server-Side JavaScript ซึ่งหมายถึง JavaScript ที่ถูกแปลทางฝั่งเซิร์ฟเวอร์ สามารถใช้ได้เฉพาะกับ LiveWire ของ Netscape โดยตรง

โปรแกรมภาษาจาวามีความสลับซับซ้อนกว่า ต้องมีการเขียนที่รัดกุมมากกว่า Java Script ถ้าเป็นงานง่ายๆ เพื่อลดภาระ เซิร์ฟเวอร์ ก็ใช้ Java Script ในการทำงาน ได้ ถ้าเป็นงานที่มีลักษณะเฉพาะ และใช้งานมากกว่า เช่น โปรแกรม Game Online ที่ function ในฝั่งผู้ใช้เยอะมาก ต้องมีการสร้าง GUI ขึ้นมา แบบนี้ก็จะต้องใช้ Java applet หรือ ActiveX

3. Introduction to Java Servlet

Server Side Application ในระยะแรก ๆ มักถูกเขียนขึ้นด้วยคอนเซ็ปของ CGI (Common Gateway Interface) โดยหลักการทำงานง่าย ๆ ก็คือ เว็บบราวเซอร์ จะทำการส่งข้อมูลที่เกิดจากการกระทำของผู้ใช้ เช่น การคลิกลิงค์หรือการกรอกแบบสอบถามไปยัง เว็บเซิร์ฟเวอร์ โดยแทนที่ เว็บเซิร์ฟเวอร์ จะทำการส่งเพ็จที่เป็น static page กลับมา เว็บเซิร์ฟเวอร์ จะทำการส่งข้อมูลดังกล่าวไปยังโปรแกรมซึ่งถูกจัดไว้ โปรแกรมดังกล่าวจะทำการประมวลผลข้อมูลที่ได้แล้วจะส่งผลกลับไปยัง เว็บเซิร์ฟเวอร์ ซึ่งทาง เว็บเซิร์ฟเวอร์ ก็จะส่งผลที่ได้นี้กลับไปยัง เว็บบราวอีกทีหนึ่ง

Server Side Application ที่เป็นผลิตผลของ CGI ในรุ่นแรก ๆ ที่ยังหลงเหลืออยู่ในปัจจุบันนี้ ยกตัวอย่าง เช่น Counter, Image Map, Guess book, Send Mail เป็นต้น จริงๆแล้วตัว CGI เองสามารถเขียนด้วยภาษาอะไรก็ได้ แต่ที่นิยมมากที่สุดเห็นจะเป็น C และ Perl อาจจะเป็นเพราะว่า CGI เป็นสิ่งที่มีมากับอินเตอร์เน็ทตั้งแต่ช่วงแรก ๆ ดังนั้นไม่ว่าจะเป็น เว็บเซิร์ฟเวอร์ ไหนก็ตาม Server Side Application พื้นฐานที่ทาง เว็บเซิร์ฟเวอร์ เหล่านั้นจะต้องสนับสนุนก็คือ CGI ซึ่งจุดนี้เองที่เป็นจุดเด่นทำให้ CGI เป็นที่นิยมใช้กันอย่างกว้างขวางจนกระทั่งปัจจุบันนี้

4. Servlet

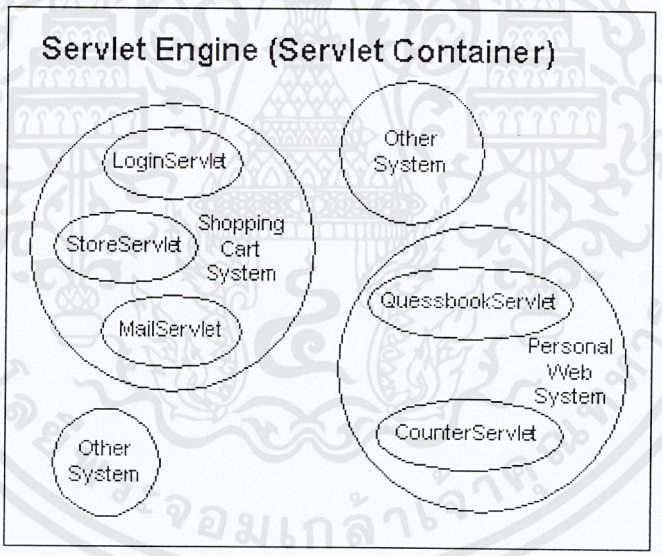
Servlet เป็น Server Side Application แบบหนึ่งซึ่งอ้างอิงคอนเซ็ปมาจาก CGI ข้อดีของ Servlet ที่อยู่เหนือ CGI อย่างแรกก็คือตัวภาษาที่ใช้เขียนซึ่งก็คือจาว่านั่นเอง จาวาเป็นภาษาที่ใช้คอนเซ็ปของ Object Oriented ในการเขียน หลายคนที่เกี่ยวข้องกับการเขียนโปรแกรมสำหรับโปรเจคใหญ่ ๆ จะทราบดีว่า Object Oriented สามารถลดความซับซ้อนของโครงสร้างโปรแกรมรวมไปถึงการอำนวยความสะดวกในการ reuse ส่วนของโปรแกรมที่เขียนไว้แล้วเพียงไร นอกจากนี้จาว่ายังเป็นภาษาที่เป็นลักษณะแบบ platform independent ซึ่งจะช่วยให้เราสามารถที่จะทำการพัฒนาระบบโดยใช้ Environment อะไรก็ได้ซึ่งโดยทั่วไปมักนิยมใช้ Window Environment โดยจะนำโปรแกรมที่เขียนเสร็จแล้วมารันบน Unix Environment เพื่อเพิ่มความเสถียรภาพของโปรแกรมแทน

นอกจากนี้ Servlet ยังมีความเร็วที่สูงกว่า CGI เพราะ Servlet ใช้หลักการของ thread โดยจะทำการสร้าง 1 thread ต่อหนึ่ง request ที่มาจาก ไคลเอนท์ ซึ่งในทางกลับกัน CGI จะทำการสร้าง 1 process ต่อหนึ่ง request ซึ่งจะ

ทำให้เปลืองทรัพยากรมากกว่าและ process ในการรันก็จะช้ากว่าด้วย ท้ายที่สุดจุดเด่นที่สำคัญของ Servlet ก็คือ API (Application Programming Interface) โดยระบบที่ทำการพัฒนาโดยใช้คอนเซ็ปของ Servlet จะสามารถเรียกใช้ API ที่ทางจาวามีมาให้ (javax.servlet.*, javax.servlet.http.*) ซึ่งจะช่วยทำให้การพัฒนาระบบดังกล่าวง่ายและเร็วยิ่งขึ้น

5. Servlet Engine

ในการรันระบบที่เขียนขึ้นโดยใช้หลักการของ Servlet เราจะต้องนำระบบดังกล่าวมาบรรจุอยู่ในสิ่ง ๆ หนึ่งที่เรียกว่า Servlet Engine ให้นึกถึง Servlet Engine คล้าย ๆ กับกล่อง ๆ หนึ่งที่ใส่ลูกปิงปองไว้หลายลูก โดยลูกปิงปองแต่ละลูกก็คือระบบ ๆ หนึ่งนั่นเอง หลายคนอาจสงสัยทำไมถึงใช้คำว่าระบบ โดยทั่วไป Server Side Application หนึ่ง ๆ ที่ถูกเขียนขึ้นโดยใช้ Servlet API จะถูกเรียกว่า Servlet ในหนึ่งระบบอาจประกอบด้วย Servlet หลายอัน ยกตัวอย่างเช่น ระบบที่เกี่ยวกับ Shopping Cart อาจจะประกอบด้วย Servlet ที่ทำหน้าที่ในการเช็คล็อกอิน , Servlet ที่ทำหน้าที่ในการเก็บข้อมูลสินค้า, Servlet ที่ทำหน้าที่ในการส่งเมล์กลับไปยังลูกค้าเพื่อบอกว่าได้ทำการส่งของไปให้แล้ว เป็นต้น ดังนั้นถ้ามองโดยรวมแล้ว Servlet Engine ก็คือที่รวมของระบบตั้งแต่หนึ่งระบบถึงหลายระบบ โดยแต่ละระบบจะประกอบด้วย Servlet หนึ่งอันหรือมากกว่า ดังรูปที่ 1.



รูปที่ 1. Servlet Engine and its Servlets

Engine เป็นเพียงกล่อง ๆ หนึ่งที่ใช้บรรจุและรันกลุ่มของ Servlet เท่านั้น ในการที่จะทำการติดต่อสื่อสารกับ ไคลเอนท์ ตัว Servlet Engine นี้จะต้องทำงานร่วมกัน เว็บเซิร์ฟเวอร์ ซึ่งเปรียบเสมือนฉากหน้าที่ติดต่อกับ ไคลเอนท์ อีกทีหนึ่ง เมื่อใดก็ตามที่มีร้องขอมาจากไคลเอนท์ ถ้าการร้องขอนั้นเจาะจงมาที่ตัว Servlet ทาง เว็บเซิร์ฟเวอร์ ก็จะทำการส่งการร้องขอนั้นมาให้ Servlet Engine ซึ่งทาง Servlet Engine ก็จะทำการเรียก Servlet ที่ไคลเอนท์ต้องการขึ้นมาทำการประมวลผลการร้องขอนั้น โดยท้ายสุด Servlet จะส่งผลกลับไปให้ Servlet Engine, Servlet Engine ก็จะส่งผลที่ได้กลับไปให้ เว็บเซิร์ฟเวอร์ ซึ่ง เว็บเซิร์ฟเวอร์ ก็จะส่งผลกลับไปให้ ไคลเอนท์

Sevlet Engine อาจจะเป็นส่วนที่ติดมากับ เว็บเซิร์ฟเวอร์ อยู่แล้วยกตัวอย่างเช่น Servlet Engine ที่อยู่ใน Netscape Enterprise Server, IBM Web Sphere หรืออาจจะเป็นส่วนที่เป็น Add-on ให้กับ เว็บเซิร์ฟเวอร์ ก็ได้เช่น Apache Jserv, Tomcat, JRun หรือแม้กระทั่งเป็นส่วนหนึ่งที่อยู่ใน Application Server เช่น BEA Weblogic เป็นต้น ทั้งนี้การเลือกใช้ Servlet Engine แต่ละชนิดก็มักขึ้นอยู่กับปัจจัยหลายอย่างเช่น ความสะดวกในการรวมระบบที่จะสร้างขึ้นมาใหม่กับระบบที่มีอยู่แล้ว, งบประมาณที่มีอยู่สำหรับโครงการหรืออาจจะรวมไปถึงทักษะและประสบการณ์ส่วนตัวของนักพัฒนาแต่ละคน

6. การทำงานของ Servlet

สมมุติว่าเรามี interface อันหนึ่งชื่อ Servlet โดย interface นี้ประกอบไปด้วยฟังก์ชันสองฟังก์ชันดัง code snippet ข้างล่างนี้

```
interface Servlet {
 public void init();
 public void service();
}
```

ถ้าเราต้องการสร้างคลาส ๆ หนึ่งชื่อ MyServlet โดยคลาสดังกล่าว implement *Servlet interface* คลาส MyServlet อาจจะเป็นอย่างข้างล่างนี้

```
class MyServlet implements Servlet {
 public void init() {
  System.out.println("Calling MyServlet init()...");
 }
 public void service() {
  System.out.println("I'm MyServlet");
 }
}
```

เราจะสังเกตเห็นว่าคลาส MyServlet ทำการ implement ฟังก์ชั่น init() และ service() ซึ่งเป็นฟังก์ชันที่ถูก define อยู่ใน Servlet interface

Servlet Engine ใช้ประโยชน์จากคอนเซ็ปของ interface ในการโหลด Servlet ต่าง ๆ ในช่วง Runtime โดย Servlet ต่าง ๆ จะถูกโหลดและ cast กลับไปเป็นอยู่ในรูปของ interface แทน

ถ้าดูจากตัวอย่างของเราหลังจาก MyServlet ถูก cast กลับไปเป็น instance ของ Servlet interface แล้วทาง Servlet Engine ก็จะสามารถเรียกฟังก์ชันอะไรก็ได้ที่ถูก define อยู่ใน Servlet interface (ในที่นี้ก็คือ init() และ

service()) ตัวอย่าง code ง่าย ๆ ที่ Servlet Engine ใช้ในการโหลด Servlet ในช่วง Runtime อาจจะเป็นอย่างข้างล่างนี้

```
public class SimpleServletEngine {
 public static void main(String args[]) throws Exception {
  if (args.length <= 0) {
   System.out.println("Usage: java SimpleServletEngine javaClassName");
   System.exit(0);
  }
  System.err.println("loading class " + args[0] + "...");
  Class cls = Class.forName(args[0]);
  Servlet servlet = (Servlet) cls.newInstance(); // casting
  servlet.init();
  servlet.service();
 }
}
```

ตัว Servlet ที่ Servlet Engine โหลดจะถูกอ้างอิงมาจาก argument ที่ชื่อ javaClassName โดย arguement นี้จะถูกส่งผ่านไปให้ Class.forName() เพื่อเรียก Class object ที่จะใช้สร้าง Servlet instance ขึ้นมาโดยใช้คำสั่ง Class.newInstance() ซึ่งหลังจากนั้น Servlet instance ที่ได้จะถูก cast กลับให้กลายเป็น instance ของ Servlet interface เพื่อ Servlet Engine จะได้ใช้ในการเรียกฟังก์ชั่น init() และ service()

เราอาจพูดง่าย ๆ ว่าหลักการทำงานของ Servlet Engine ก็คือการโหลดคลาสใดคลาสหนึ่งขึ้นมาโดยคลาสนั้นจะต้อง implement ตัว interface ที่ชื่อ Servlet ซึ่งทาง Servlet Engine ทราบอยู่แล้วว่ามีฟังก์ชันอะไรประกอบอยู่ใน Servlet interface บ้างโดยถ้า Service Engine ทำการ cast คลาสที่ implement Servlet interface (ในที่นี้ก็คือ MyServlet) กลับไปเป็น instance ของ Servlet interface แล้วตัว Servlet Engine ก็จะสามารถเรียกฟังก์ชันที่ชื่อ init() และ service() ที่ define อยู่ใน Servlet interface จากคลาสดังกล่าวได้

ในกรณีที่เราไม่ต้องการ implement ฟังก์ชันทุกฟังก์ชันที่ถูก define อยู่ใน Servlet interface เราก็อาจจะสร้างคลาสที่เป็น abstract ขึ้นมาคลาสหนึ่งก่อนโดยคลาสนี้จะทำการ implement ฟังก์ชันบางฟังก์ชันที่อยู่ใน Servlet interface แล้วให้ subclass ของ abstract คลาสนี้ทำการ implement ฟังก์ชันที่เหลือ ยกตัวอย่างเช่น

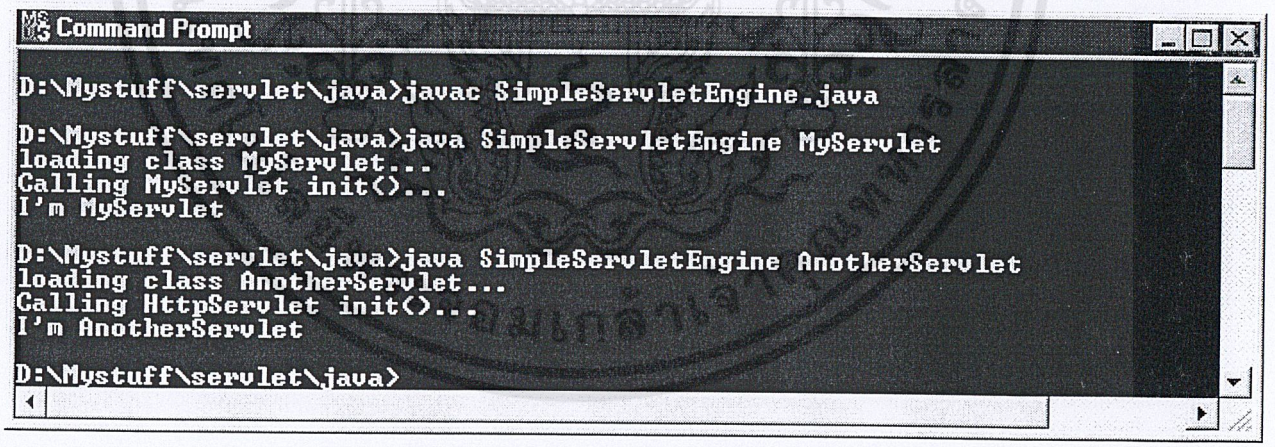
```
abstract class HttpServlet implements Servlet {
 public void init() {
  System.out.println("Calling HttpServlet init()...");
 }
```

```
// not implemented yet
public abstract void service();
}
```

ถ้าเราต้องการสร้าง Servlet ขึ้นมาโดยจะ implement แค่ฟังก์ชัน service() อย่างเดียว เราก็เพียง subclass คลาส HttpServlet เท่านั้น โดยคลาส Servlet ของเราอาจจะเป็นดังตัวอย่างข้างล่างนี้

```
class AnotherServlet extends HttpServlet {
public void service() {
 System.out.println("I'm AnotherServlet");
}
}
```

เราจะสังเกตเห็นว่าคลาส AnotherServlet จะไม่ต้องทำการ implement ฟังก์ชัน init() เพราะตัว parent คลาสของมันซึ่งก็คือ HttpServlet ได้ทำการ implement ไปแล้ว ในการโหลดคลาส AnotherServlet เข้าไปยัง Servlet Engine ก็ยังใช้หลักการเดียวกันกับคลาส MyServlet ข้างต้นคือโหลดคลาส AnotherServlet แล้ว cast กลับให้กลายเป็น instance ของ Servlet interface ซึ่งผลจากการรันโปรแกรม SimpleServletEngine โดยโหลด Servlet (หรือคลาส)ที่ชื่อ MyServlet และ AnotherServlet ก็จะออกมาเป็นดังรูป



รูปที่ 2. การโหลด Servlet ต่าง ๆ ของ SimpleServletEngine

ภาคผนวก ข.

การติดตั้งเว็บไซต์ระบบให้บริการแผนที่บนเว็บ

1. การตั้งค่าเว็บเซิร์ฟเวอร์

โปรเว็บเป็นโปรแกรมที่ทำงานบนระบบปฏิบัติการวินโดว์ (Microsoft Windows) ดังนั้นเราจึงเลือกใช้ Apache Web Server for Windows โดยดาวน์โหลดได้จาก

http://www.apache.org/dist/httpd/binaries/win32/

การตั้งค่าสำหรับโปรแกรม Apache จะสามารถแก้ค่าของเซิร์ฟเวอร์โดยการเข้าไปแก้ในไฟล์ httpd.conf ในไดเรคทอรี่ binโดยจะต้องเพิ่มคำสั่งเหล่านี้ลงไป เพื่อกำหนดพาทของโปรเว็บและพาทต่างๆที่จำเป็นในการใช้งานโดย C:/WIN-PROLOG4040/PROWEB เป็นตำแหน่งของโปรเว็บโดยที่โปรเว็บต้องอยู่ในไดเรคทอรี่ของ WIN-PROLOG ด้วย

```
Alias /images "C:/WIN-PROLOG4040/PROWEB/IMAGES"
<Directory "C:/WIN-PROLOG4040/PROWEB/IMAGES">
    Options Indexes FollowSymLinks MultiViews IncludesNoExec
    AddOutputFilter Includes html
    AllowOverride None
    Order allow,deny
    Allow from all
</Directory>

Alias /map "C:/WIN-PROLOG4040/PROWEB/HTML"
<Directory "C:/WIN-PROLOG4040/PROWEB/HTML">
    Options Indexes FollowSymLinks MultiViews IncludesNoExec
    AddOutputFilter Includes html
    AllowOverride None
    Order allow,deny
    Allow from all
</Directory>

ScriptAlias /proweb/ "C:/WIN-PROLOG4040/PROWEB/"
ScriptAlias /ProWeb/ "C:/WIN-PROLOG4040/PROWEB/"
<Directory "C:/WIN-PROLOG4040/PROWEB">
```

AllowOverride None

Options None

Order allow,deny

Allow from all

</Directory>

2. การตั้งค่าโปรแกรมโปรเว็บ

1. คัดลอกไฟล์ pro386w.exe จากโปรแกรม WIN-PROLOG ลงไปในของไดเรคทอรี่โปรเว็บ พร้อมทั้งเปลี่ยนชื่อไฟล์เป็น proweb.sys
2. คัดลอกไฟล์ DBLINK.PC ในไดเรคทอรี่ system ที่อยู่ในโปรแกรมโปรล็อกลงไปที่ไดเรคทอรี่ proweb\system
3. คัดลอกไฟล์ LPADBW.DLL ในไดเรคทอรี่โปรแกรมโปรล็อกลงไปที่ไดเรคทอรี่ proweb
4. คัดลอกไฟล์โปรล็อก(นามสกุล pl)ทั้งหมดไว้ใน proweb/examples
5. คัดลอกไฟล์ HTML ทั้งหมดไว้ใน proweb/html
6. คัดลอกไฟล์รูปภาพทั้งหมดไว้ใน proweb/images
7. เมื่อทำตามขั้นตอนนี้เสร็จแล้ว ให้เรียกไปที่ URL http://localhost/map/index.html

บรรณานุกรม

[1] ประภากร ลาภประสพ วิศิษฏ์ หิรัญกิตติ "**ระบบนำร่องรถยนต์ชาญฉลาด**" การประชุมวิชาการและวิศวกรรมคอมพิวเตอร์แห่งชาติ (NSCEC 2000) มหาวิทยาลัยเชียงใหม่ เชียงใหม่ 2544.

[2] พยนต์ ทิมเจริญ "**ระบบแผนที่ผลิตโดยแผนที่ทหาร**" กรุงเทพฯ 2544.
http://www.rtsd.mi.th/publication/toponymic/toponymicguideline.zip

[3] ศาสตราจารย์ วริทธิ์ อึ้งภากรณ์ "เทอร์โบโปรล็อกและระบบเชี่ยวชาญ" ฟิสิกส์เซ็นเตอร์การพิมพ์ กรุงเทพฯ 2535.

[4] Clocksin W.F. Mellish C.S., "**Programming in Prolog**", Springer-Verlag, 1987.

[5] Hall M. "**Core Servlet and JavaServer Pages**", Prentice Hall PTR, 2000.

[6] WIN-PROLOG Programming Guide, Logic Programming Associates Ltd., 2000.,
http://www.lpa.co.uk/ftp/4300/win_prg.pdf

[7] "**ProWeb Server User Guide**", Logic Programming Associates Ltd., 2000.,
http://www.lpa.co.uk/ftp/4300/pws_ref.pdf

[8] "**ProData Interface**", Logic Programming Associates Ltd., 2000.,
http://www.lpa.co.uk/ftp/4300/pdi_ref.pdf

[9] Albrecht Kleine, **WmfView**, 2002,
http://www.sax.de/~adlibit/