บทที่ 8
การวางแผน
(Planning)

การวางแผนคืองานเกี่ยวกับการค้นหาลำดับการกระทำต่าง ๆ ที่เป็นไปเพื่อให้บรรลุเป้าหมาย เอเยนต์ที่แก้ปัญหาโดยอาศัยเทคนิคค้นหาตามที่กล่าวมาแล้วในบทที่ 3-4 ต่างก็เป็นตัวอย่างของเอเยนต์ที่กำลังวางแผนทั้งสิ้น

8.1 ปัญหาการวางแผนทั่วไป

ในบทที่ 3 และ 4 นั้น เอเยนต์แก้ปัญหา (Problem solving agent) ทำงานโดยค้นหาไปใน State space โดยที่สถานะหรือโนดต่าง ๆ แทนสถานการณ์ในขณะหนึ่ง ๆ การค้นหาไม่ว่าจะนำข้อมูลหรือความรู้มาใช้ประกอบหรือไม่ก็ตาม ต่างก็เริ่มต้นที่สถานะเริ่มต้น แล้วดำเนินไปจนกระทั่งพบสถานะเป้าหมาย (ในกรณีที่หาคำตอบได้) จะเห็นว่าสถานะใน State space ต่างก็สามารถเป็นตัวแทนของสถานการณ์ของปัญหาได้ทั้งสิ้น เช่น สถานะของปัญหา 8-puzzle แสดงถึงสภาพปัญหาในขณะใดขณะหนึ่งได้อย่างสมบูรณ์ ส่วนแอคชันที่กระทำได้ของ 8-puzzle ก็บรรยายถึงวิธีเปลี่ยนสถานะได้ครบถ้วน แต่ยังมีปัญหาอีกมากที่แก้ไขได้ยาก อุปสรรคที่เห็นได้ชัดเจนและมักเกิดขึ้นบ่อยกับปัญหาทั่วไปได้แก่

1. เอเยนต์มีจำนวนแอคชันให้เลือกกระทำเป็นจำนวนมาก แต่ส่วนใหญ่เป็นการกระทำที่ไม่ทำให้บรรลุเป้าหมาย ตัวอย่างเช่น ในการสั่งซื้อหนังสือออนไลน์ ไม่ต้องบอกชื่อเรื่องหรือชื่อผู้แต่ง แต่ระบุหมายเลข ISBN ของหนังสือ หมายเลข ISBN เป็นเลข 10 ตัว ถ้าแอคชันของการซื้อหนังสือ (Buy) โดยระบุหมายเลข ISBN ของหนังสือมีจำนวน 1 แอคชันต่อ 1 หมายเลข ISBN แอคชันทั้งหมดในการซื้อหนังสือที่อาจเป็นไปได้จะมีจำนวน = 10^{10} = 10,000 ล้าน แอคชัน เอเยนต์ต้องการซื้อหนังสือที่หมายเลข ISBN0137903952 ต้องใช้อัลกอริทึมของการค้นหาทำการค้นหาสถานะหมื่นล้านสถานะไปจนกว่าจะพบสถานะที่ตรงตามเป้าหมาย ซึ่งมีอยู่เป็นจำนวน 1 ในหมื่นล้านสถานะ แต่ถ้าเอเยนต์ฉลาดขึ้น ควรหาทางทำงานย้อนจากเป้าหมายสุดท้าย นั่นคือ Have(ISBN0137903952) และสร้างแอคชัน Buy(ISBN0137903952) ขึ้นตรง ๆ แต่ในการทำเช่นนี้ เอเยนต์ต้องมีความรู้ทั่วไปก่อนว่า Buy(x) ก่อให้เกิดผล Have(x)

2. เอเยนต์ขาดฮิวริสติกฟังก์ชันที่ดี สมมติว่าเป้าหมายของเอเยนต์คือต้องการซื้อหนังสือ 4 เล่มแตกต่างกันแบบออนไลน์ ก็จะมีแผนจำนวน 10^{40} แผนสำหรับงาน 4 ขั้นตอน (ซื้อหนังสือ 1 เล่มใช้ 1 ขั้นตอน) การค้นหาโดยไม่มีฮิวริสติกช่วยเป็นเรื่องที่ทำไม่ได้เลยในปัญหานี้ สำหรับมนุษย์แล้ว อาจมองเห็นได้ชัดเจนว่า ฮิวริสติกของปัญหานี้คือ ค่าใช้จ่ายของแต่ละสถานะคิดจากจำนวนหนังสือที่ยังไม่ได้ซื้อ แต่การมองเห็นฮิวริสติกเช่นนี้เป็นความสามารถของมนุษย์ เอเยนต์มองไม่เห็น เพราะสำหรับเอเยนต์แล้ว การทำ Goal test ก็เหมือนกล่องดำที่จะให้ค่าเป็นจริง หรือเท็จในแต่ละสถานะเท่านั้น มนุษย์จึงต้องเป็นผู้กำหนดฮิวริสติกให้กับปัญหาแต่ละปัญหา แต่ถ้าเอเยนต์ประเภทวางแผนรู้ว่าเป้าหมายประกาศไว้เป็นเป้าหมายย่อยจำนวนหนึ่งที่เชื่อมกันเป็น Conjunction เอเยนต์จะสามารถใช้ฮิวริสติกง่าย ๆ เช่น ใช้จำนวน Conjunct ที่ยังไม่บรรลุเป้าหมายมาเป็นค่าใช้จ่ายของสถานะ เช่นปัญหาการซื้อหนังสือดังกล่าว เป้าหมายจะเป็น Have(A) $\wedge$ Have(B) $\wedge$ Have(C) $\wedge$ Have(D) ถ้าสถานะหนึ่งประกอบด้วย Have(A) $\wedge$ Have(C) แล้ว สถานะนั้นมีค่าเป็น 2 เอเยนต์ที่ใช้วิธีนี้จะมีฮิวริสติกที่เหมาะสมได้เองโดยอัตโนมัติ

จากที่กล่าวมา ได้กล่าวถึงคำว่าปัญหาย่อย ทั้งนี้เนื่องจากมีปัญหาบางประเภทที่ซับซ้อนมาก จำเป็นต้องแบ่งปัญหาออกเป็นส่วนย่อยก่อน แล้วจึงทำงานกับส่วนย่อยแต่ละส่วน เมื่อได้คำตอบย่อยแล้วนำมารวมกัน จะทำให้ได้คำตอบสำหรับปัญหาทั้งข้อ แต่ถ้าไม่ทำเช่นนี้ จำนวนการรวมสถานะ (Combination ของสถานะ) ในปัญหาจะมีขนาดใหญ่มากจนเกินกว่าจะจัดการได้ทั้งทางด้านเวลา และหน่วยความจำ

การแยกปัญหาย่อยมีสิ่งสำคัญที่ต้องคำนึงอยู่เสมอ 2 ข้อได้แก่

1. ต้องไม่ทำให้เกิดการคำนวณหาสถานะในปัญหาใหม่ทั้งหมด หลังการเปลี่ยนสถานะแต่ละครั้ง โดยพิจารณาเฉพาะสถานะเพียงบางส่วนเท่านั้นที่อาจจะเปลี่ยนไป ตัวอย่างเช่น ถ้าจะย้ายจากห้องหนึ่งไปยังอีกห้องหนึ่ง สิ่งที่ไม่ได้รับผลกระทบจากการย้ายได้แก่ ตำแหน่งของประตู หน้าต่างในห้องทั้ง 2 ห้อง ดังนั้นจึงต้องตัดสินให้ดีว่าเมื่อเปลี่ยนสถานะแต่ละครั้ง มีอะไรเปลี่ยน และมีอะไรบ้างที่ไม่เปลี่ยน ปัญหา 8-puzzle เห็นได้ง่ายว่าสถานะแต่ละสถานะเปลี่ยนแปลงไปอย่างไร หลังการขยับแต่ละครั้ง การบันทึกความเปลี่ยนแปลงก็ทำได้โดยไม่สิ้นเปลืองมากนัก (ปริมาณการทำงานมีไม่มาก) บรรยายกฎการเปลี่ยนสถานะของกระดาน 8-puzzle ของการเดินแต่ละตาได้ง่ายและชัดเจน

เมื่อพิจารณาปัญหาการนำทางหุ่นยนต์ให้เดิน (เคลื่อนไหว) ไปรอบบ้าน สถานการณ์มีความซับซ้อนมากขึ้น การบรรยายสถานะ 1 สถานะเป็นเรื่องใหญ่ เพราะต้องกล่าวถึงตำแหน่งของสิ่งของต่าง ๆ ในบ้าน พร้อมกับตำแหน่งของหุ่นยนต์ แต่การที่หุ่นยนต์มีการกระทำอย่างใด

อย่างหนึ่ง จะส่งผลเพียงส่วนน้อยกับสถานะทั้งสถานะ (ซึ่งมีขนาดใหญ่) เช่น ถ้าหุ่นยนต์ผลักโต๊ะเลื่อนออกจากที่แล้ว สิ่งที่เปลี่ยนไปคือ ตำแหน่งของโต๊ะ และสิ่งของที่อยู่บนโต๊ะ แต่วัตถุอื่นในบ้านไม่มีการเปลี่ยนแปลง ดังนั้นเมื่อเปลี่ยนจากสถานะหนึ่งไปเป็นอีกสถานะหนึ่ง แทนที่จะเขียนกฎบรรยายการเปลี่ยนแปลงสถานะทั้งสถานะ เราจะเขียนกฎที่บรรยายเฉพาะบางส่วนของสถานะที่ได้รับผลกระทบ ที่เหลือนอกจากนั้นไม่มีการเปลี่ยนแปลง คงปล่อยให้เป็นเช่นเดิม

2. หาทางแบ่งปัญหายากออกเป็นปัญหาย่อยที่ง่ายขึ้นหลาย ๆ ข้อ แต่การพยายามแยกปัญหาย่อย บางครั้งอาจจะเป็นไปไม่ได้หรือทำได้ยาก ตัวอย่างเช่น ปัญหาการย้ายเครื่องเรือนออกไปจากห้อง อาจจะทำโดยแยกตัวปัญหาออกเป็นปัญหาเล็ก ๆ หลายปัญหา แต่ละปัญหาคือการย้ายเครื่องเรือนแต่ละชิ้นออกไปจากห้อง ปัญหาย่อยแต่ละข้อสามารถนำมาพิจารณาเพิ่มเติม เช่น ถือว่าการย้ายลิ้นชัก เป็นการย้ายพิเศษที่แตกต่างออกไปจากเครื่องเรือนชิ้นอื่น ถ้ามีตู้หนังสืออยู่หลังโซฟา ก็ต้องย้ายโซฟาออกก่อนที่จะย้ายตู้หนังสือ การแก้ปัญหาเหล่านี้ จึงต้องมีวิธีที่ทำให้สามารถทำงานกับปัญหาย่อยแต่ละปัญหาได้แตกต่างกัน เพื่อให้จัดการกับปัญหาหลายรูปแบบได้อย่างเหมาะสม ปัญหาประเภทนี้เรียกว่า ปัญหาที่เกือบกระจาย (Nearly decomposable) คือปัญหาที่ยังไม่สามารถกระจายแยกย่อยได้สมบูรณ์แบบ ส่วนที่กระจายไปแล้วยังมีส่วนที่อิงกับปัญหาข้ออื่น และส่วนที่กระจายแยกย่อยไปแล้วอาจจะไม่ทำให้สถานการณ์ดีขึ้นด้วยซ้ำ ถ้าการแก้ปัญหาย่อยนั้นไปขัดแย้งกับปัญหาย่อยอื่น แต่ถ้าสามารถแยกแยะออกเป็นปัญหาย่อยจำนวนหนึ่งได้สำเร็จ เท่ากับมีเป้าหมายย่อย (Subgoal) หลายเป้าหมาย ซึ่งเป้าหมายเหล่านี้ต่างก็เป็นอิสระต่อกัน แต่อาจจะต้องทำงานเพิ่มมากขึ้นในการรวมแผนย่อยหลังจากนี้

8.2 ภาษาที่ใช้ในปัญหาการวางแผน

ปัญหาการวางแผนที่กล่าวถึงแล้วนี้มีการแทนด้วยสถานะ แอคชัน และเป้าหมาย ตามลักษณะของเอเยนต์ทั่วไป สิ่งเหล่านี้ทำให้ใช้อัลกอริทึมในการวางแผนได้เพราะทำให้ปัญหามีโครงสร้างทางตรรกะ แต่ก็จำเป็นต้องมีภาษาที่ใช้จัดการกับงานวางแผนที่แตกต่างออกไปจากตรรกะแบบเดิม ภาษานี้ต้องสามารถแสดงหรือบรรยายความหลากหลายต่าง ๆ ของตัวปัญหา และต้องมีความสามารถเฉพาะตัวเพียงพอที่จะนำมาใช้กับอัลกอริทึมที่มีประสิทธิภาพได้ด้วย ในที่นี้จะใช้ภาษา STRIPS ซึ่งเป็นภาษาสำหรับการวางแผน ชื่อภาษามาจากคำว่า Stanford Research Institute Problem Solver

8.2.1 การแทนค่า (Representation)

การแทน (Representation) ในภาษา STRIPS แบ่งออกเป็น

1. การแทนสถานะ โดยแยกความเป็นไปในโลกให้อยู่ในสภาพเดี่ยว ๆ แต่ละเรื่องแทนด้วยค่าข้อความบวก เขียนเป็นตรรกะ แล้วนำมาเชื่อมรวมกันเป็น Conjunction เช่นตัวอย่างการเขียนข้อความแบบตรรกศาสตร์ประพจน์เพื่อแทนสถานะของเอเยนต์บุคคลที่ไม่มีความสุขอาจจะใช้คำว่า Poor $\wedge$ Unknown หรืออีกตัวอย่างหนึ่งได้แก่การเขียนข้อความแบบตรรกศาสตร์อันดับหนึ่งว่า

$$At(Plane_1 , Melbourne) \wedge At(Plane_2 , Sydney)$$

เพื่อแทนสถานะในปัญหาการส่งสินค้า ค่าข้อความที่บรรยายสถานะแบบอันดับหนึ่งนี้จะต้องไม่มีตัวแปร ไม่มีฟังก์ชัน ค่าข้อความที่มีตัวแปรเช่น At(x, y) หรือ At(Father(Fred), Sydney) ใช้ไม่ได้ และใช้สมมติฐานโลกปิด (Closed-world assumption) นั่นคือสภาพใดที่ไม่เคยกล่าวถึงในสถานะมาก่อนเลยถือว่าไม่มี (หรือเป็นเท็จ)

2. การแทนเป้าหมาย เป้าหมายเป็นสถานะที่ระบุไว้เป็นส่วน ๆ แทนได้ด้วย Conjunction ของค่าข้อความบวกที่ไม่มีตัวแปร เช่น Rich $\wedge$ Famous หรือ $At(P_2 , Tahiti)$ เรากล่าวว่า สถานะ s บรรลุเป้าหมาย g ถ้า s ประกอบด้วยทุกเทอมในเป้าหมาย g (อาจจะเกินได้) เช่น สถานะ Rich $\wedge$ Famous $\wedge$ Miserable บรรลุเป้าหมาย Rich $\wedge$ Famous

3. การแทนแอคชัน แอคชันอยู่ในเทอมของเงื่อนไขก่อนหน้า (Precondition) ซึ่งต้องเป็นจริงก่อนที่จะมีการดำเนินงาน และผลที่เกิดหลังจากการดำเนินงานนั้นแล้ว เช่น แอคชันของเครื่องบินที่บินจากที่หนึ่งไปยังอีกที่หนึ่ง เขียนได้ดังนี้

Action(Fly(p, from, to),

PRECOND: At(p, from) $\wedge$ Plane(p) $\wedge$ Airport(from) $\wedge$ Airport(to)

EFFECT: $\neg$At(p, from) $\wedge$ At(p, to))

การเขียนด้วยรูปแบบนี้เรียกว่า (Action schema) สามารถใช้แทนแอคชันการบินได้หลากหลายแอคชัน โดยแทนค่าตัวแปร p, from และ to ด้วยค่าคงที่ต่าง ๆ โดยทั่วไปแล้ว action schema ประกอบด้วย 3 ส่วนคือ

1. ชื่อของแอคชัน และรายการพารามิเตอร์ เช่น Fly(p, from, to) ใช้ระบุตัวแอคชัน

2. ส่วนเงื่อนไขก่อนหน้า (Precondition) เป็น Conjunction ของค่าข้อความบวก (Positive literal) ที่ไม่เป็นฟังก์ชัน บอกให้รู้ว่าสิ่งเหล่านี้ต้องเป็นจริงในสถานะก่อนการดำเนินงาน (Execute) ตัวแปรใด ๆ ที่ปรากฏในค่าข้อความส่วนนี้ต้องมีอยู่แล้วในรายการพารามิเตอร์

3. ส่วนผล (Effect) เป็น Conjunction ของค่าข้อความที่ไม่เป็นฟังก์ชัน บอกถึงผลที่เกิดขึ้นหลังการดำเนินงาน และค่าข้อความบวก P ใด ๆ ที่อยู่ในส่วนผลจะต้องมีค่าความจริงเป็นจริง และปรากฏอยู่ในสถานะที่เป็นผลมาจากการทำแอคชัน ในขณะที่ค่าข้อความนิเสธ ¬P ต้องเป็นเท็จ ตัวแปรใด ๆ ที่ปรากฏในส่วนผลต้องมีอยู่แล้วในรายการพารามิเตอร์

ระบบวางแผนบางระบบใช้คำว่า Add list สำหรับค่าข้อความบวก และใช้ Delete list สำหรับค่าข้อความนิเสธ

คำตอบของปัญหาการวางแผนคือลำดับของแอคชันซึ่งเมื่อนำมาดำเนินการตั้งแต่สถานะเริ่มต้น จะได้ผลลัพธ์เป็นสถานะที่บรรลุเป้าหมายได้

ตารางที่ 8.1 การเปรียบเทียบภาษา STRIPS และภาษา ADL

ภาษา STRIPS	**ภาษา ADL**
ใช้ค่าข้อความบวกเท่านั้นภายในสถานะ เช่น Poor ∧ Unknown	ใช้ค่าข้อความทั้งบวกและนิเสธในสถานะ เช่น ¬Rich ∧ ¬Famous
ใช้สมมติฐานโลกปิด ค่าข้อความที่ไม่เคยอ้างถึงมาก่อนจะเป็นเท็จ	ใช้สมมติฐานโลกเปิด ค่าข้อความที่ไม่เคยอ้างถึงมาก่อนเป็น unknown
ส่วนผล P ∧ ¬Q หมายถึงเพิ่ม P ลบ Q	ส่วนผล P ∧ ¬Q หมายถึงเพิ่ม P และ ¬Q และลบ ¬ P และ Q
ในเป้าหมายมีแต่ค่าข้อความที่ไม่มีตัวแปร เช่น Rich ∧ Famous	มีตัวแปรและตัวบ่งปริมาณในเป้าหมาย เช่น $\exists x$ At(P_1, x) ∧ At(P_2, x) คือเป้าหมายว่า มี P_1 และ P_2 อยู่ในที่เดียวกัน
เป้าหมายเป็น Conjunction เช่น Rich ∧ Famous	เป้าหมายมีทั้ง Conjunction และ Disjunction ¬ Poor ∧ (Famous∨ Smart)
ส่วนผลเป็น Conjunction	มีผลลัพธ์เป็นเงื่อนไข เช่น **When** P: E หมายความว่า E จะเป็นผล ถ้า P เป็นจริง
ไม่มีการเท่ากัน	มีเพรดิเคต =
ไม่มีการกำหนดชนิด	มีชนิดของตัวแปร เช่น (p : Plane)

ภาษา STRIPS พยายามเพิ่มข้อจำกัดในตัวภาษาเพื่อให้สามารถนำมาใช้กับอัลกอริทึมของการวางแผนได้ดีมีประสิทธิภาพมากขึ้น แต่ภาษานี้ก็ยังไม่มีความสามารถพอที่จะใช้ถ่ายทอดให้เห็นลักษณะของปัญหาจริงได้หลายเรื่อง จึงมีการพัฒนาภาษาใหม่ขึ้นหลายภาษา ภาษา ADL (Action Description Language) เป็นภาษาสำคัญภาษาหนึ่ง เปรียบเทียบกับภาษา STRIPS ได้ตามตารางที่ 8.1

แอคชัน Fly ในภาษา ADL เขียนดังนี้

Action(Fly(p : Plane, from : Airport, to : Airport),

PRECOND: At(p, from) $\wedge$ (from $\neq$ to)

EFFECT: $\neg$At(p, from) $\wedge$ At(p, to)).

บรรทัดแรกมีการบรรยาย p : Plane ในรายการพารามิเตอร์เป็นเหมือนกับการเขียนย่อของเพรดิเคท Plane(p) เขียนเช่นนี้ทำให้อ่านง่ายขึ้น เงื่อนไขก่อนหน้า from $\neq$ to มีไว้เพื่อแสดงว่าการบินไม่สามารถบินกลับมายังสนามบินที่บินขึ้นได้ ลักษณะการบรรยายเช่นนี้ไม่มีในภาษา STRIPS

Init(At(C_1, CHM) $\wedge$ At(C_2, BKK) $\wedge$ At(P_1, CHM) $\wedge$ At(P_2, BKK)
$\wedge$ Cargo(C_1) $\wedge$ Cargo(C_2) $\wedge$ Plane(P_1) $\wedge$ Plane(P_2)
$\wedge$ Airport(BKK) $\wedge$ Airport(CHM))
Goal(At(C_1, BKK) $\wedge$ At(C_2, CHM))
Action(Load(c, p, a) ,
PRECOND: At(c, a) $\wedge$ At(p,a) $\wedge$ Cargo(c) $\wedge$ Plane(p) $\wedge$ Airport(a)
EFFECT: $\neg$ At(c, a) $\wedge$ In(c, p))
Action(Unload(c, p, a) ,
PRECOND: In(c, p) $\wedge$ At(p,a) $\wedge$ Cargo(c) $\wedge$ Plane(p) $\wedge$ Airport(a)
EFFECT: At(c, a) $\wedge$ $\neg$In(c, p))
Action(Fly(p, from, to) ,
PRECOND: At(p, from) $\wedge$ Plane(p) $\wedge$ Airport(from) $\wedge$ Airport(to)
EFFECT: $\neg$At(p, from) $\wedge$ At(p, to))

รูปที่ 8.1 ปัญหาขนส่งสินค้าทางอากาศ ใช้ภาษา STRIPS

8.2.2 ตัวอย่างการขนส่งสิ่งของทางอากาศ (Air cargo transport)

ปัญหาการขนส่งสิ่งของทางอากาศ เกี่ยวข้องกับการขนของขึ้น (Load) และขนของลง (Unload) จากเครื่องบิน และนำสิ่งของเหล่านี้บินไปส่งยังที่ต่าง ๆ สามารถกำหนดปัญหาได้ 3 แอคชัน คือ Load, Unload, และ Fly การทำแอคชันเหล่านี้ต้องทำหลังจากเพรดิเคท In(c, p) ซึ่งหมายความว่าสิ่งของ (c) อยู่ในเครื่องบิน p และ At(x,a) หมายความว่าอ็อบเจ็กต์ x (ซึ่งอาจจะหมายถึงเครื่องบิน หรือสิ่งของก็ได้) อยู่ที่สนามบิน a เขียนปัญหาได้ดังรูปที่ 8.1

แผนที่เป็นคำตอบของปัญหานี้ได้แก่

[Load(C_1, P_1, CHM) , Fly(P_1, CHM, BKK),

Load(C_2, P_2, BKK) , Fly(P_2, BKK, CHM)]

Init(At(Flat, Axle) ∧ At(Spare, Trunk))

Goal(At(Spare, Axle))

Action(Remove(Spare, Trunk),

PRECOND: At(Spare, Trunk)

EFFECT: ¬ At(Spare, Trunk) ∧ At(Spare, Ground))

Action(Remove(Flat, Axle),

PRECOND: At(Flat, Axle)

EFFECT: ¬ At(Flat, Axle) ∧ At(Flat, Ground))

Action(PutOn(Spare, Axle),

PRECOND: At(Spare, Ground) ∧ ¬ At(Flat, Axle)

EFFECT: ¬ At(Spare, Ground) ∧ At(Spare, Axle))

Action(LeaveOvernight,

PRECOND: At(Spare, Ground) ∧ ¬ At(Flat, Axle)

EFFECT: ¬ At(Spare, Ground)∧¬ At(Spare, Axle)∧¬ At(Spare, Trunk)

∧¬ At(Flat, Ground) ∧ ¬ At(Flat, Axle)

รูปที่ 8.2 ปัญหายางแบน ใช้ภาษา ADL

8.2.3 ตัวอย่างปัญหายางแบน

ในปัญหาการเปลี่ยนยางเมื่อยางแบน เป้าหมายได้แก่การนำยางสำรองไปติดตั้งที่แกนล้อให้ถูกต้องเหมาะสม ส่วนสถานะเริ่มต้นได้แก่การมียางแบนติดอยู่ที่แกนล้อ ส่วนยางสำรองอยู่ในกระโปรงหลัง สำหรับปัญหานี้จะพิจารณาอย่างง่าย ๆ ไม่ซับซ้อน มีแอคชันให้กระทำ 4 แอคชัน คือ นำยางสำรองออกจากกระโปรงหลัง นำยางที่แบนออกจากแกนล้อ ใส่ยางสำรองที่แกนล้อ และทิ้งรถไว้ทั้งคืนไม่ทำอะไรเลย สมมติว่ารถอยู่ในเขตไม่ปลอดภัย ถ้าทิ้งรถไว้ทั้งคืนจะมีผลให้ยางล้อรถหาย

ใช้ภาษา ADL บรรยายตัวปัญหาได้ดังรูปที่ 8.2 สังเกตได้ว่าการบรรยายนี้ใช้ประพจน์ล้วน และส่วนเงื่อนไขก่อนหน้ามีนิเสธอยู่ด้วย ซึ่งต่างจากวิธีการของภาษา STRIPS

8.2.4 ตัวอย่างปัญหา Blocks world

ปัญหา Blocks world เป็นปัญหาที่มีชื่อเสียงที่สุดที่นำมาใช้ในการวางแผน โดเมนของปัญหาประกอบด้วยกล่องสี่เหลี่ยมวางอยู่บนโต๊ะ กล่องสามารถวางซ้อนเป็นตั้งได้ แต่ต้องวางซ้อนทีละกล่อง มีแขนกลยื่นมาหยิบกล่องและย้ายตำแหน่งของกล่องได้ 2 แบบ คือ ย้ายกล่องหนึ่งไปวางบนอีกกล่องหนึ่ง หรือย้ายกล่องไปวางบนโต๊ะ แขนนี้หยิบได้ครั้งละ 1 กล่องเท่านั้น จึงไม่สามารถหยิบกล่องที่มีกล่องวางซ้อนอยู่ได้ เป้าหมายของปัญหา Blocks world มักจะเป็นการเรียงกล่องเป็นตั้ง จำนวนตั้งแต่ 1 ตั้งขึ้นไป ตัวอย่างเช่น ต้องการให้กล่อง A อยู่บนกล่อง B และให้กล่อง C อยู่บนกล่อง D

กำหนดให้

On(b, x) หมายถึงกล่อง b อยู่บน x เมื่อ x เป็นกล่องอื่น หรือเป็นพื้นโต๊ะ

Move(b, x, y) หมายถึงการย้ายกล่อง b จากบน x ไปไว้บน y

เงื่อนไขก่อนหน้าของการย้ายกล่อง b คือต้องไม่มีกล่องอยู่บนกล่อง b นั่นคือ

$\neg\exists x$ On(x, b) หรือ $\forall x \neg$ On(x, b)

แต่ถ้าใช้ภาษา STRIPS จะใช้เพรดิเคทใหม่คือ Clear(x) หมายถึงไม่มีอะไรอยู่บน x

การใช้ Move ย้ายกล่อง b จาก x ไป y ถ้าทั้ง b และ y ว่าง หลังจากย้ายแล้ว x จะว่าง แต่บน y ไม่ว่างแล้ว เขียนบรรยายเป็น ภาษา STRIPS ว่า

Action(Move(b, x, y) ,

PRECOND: On(b, x) $\wedge$ Clear(b) $\wedge$ Clear(y) ,

EFFECT: On(b, y) $\wedge$ Clear(x) $\wedge \neg$ On(b, x) $\wedge \neg$ Clear(y))

แอคชันนี้ใช้ไม่ได้เมื่อ x และ y ต่างก็เป็นพื้นโต๊ะ เมื่อ x=Table แอคชันนี้ให้ผลเป็น Clear(Table) ซึ่งไม่จริง เพราะบนโต๊ะต้องไม่ว่าง โต๊ะไม่จำเป็นต้องใช้ Clear จึงต้องเพิ่มแอคชันใหม่เพื่อย้ายกล่อง b จาก x ไปวางบนโต๊ะ ดังนี้

เมื่อปรับปรุงเพิ่มเติมแล้ว ดูการเขียนปัญหาได้ในรูปที่ 8.3

Init(On(A,Table) ∧ On(B, Table) ∧ On(C, Table)
∧ Block(A) ∧ Block(B) ∧ Block(C)
∧ Clear(A) ∧ Clear(B) ∧ Clear(C))
Goal(On(A, B) ∧ On(B, C))
Action(Move(b, x, y) ,
PRECOND: On(b, x) ∧ Clear(b) ∧ Clear(y) ∧ Block(b) ∧
(b ≠ x) ∧ (b ≠ y) ∧ (x ≠ y),
EFFECT: On(b, y) ∧ Clear(x) ∧ ¬ On(b, x) ∧ ¬ Clear(y))
Action(MoveToTable(b, x),
PRECOND: On(b, x) ∧ Clear(b) ∧ Block(b) ∧ (b ≠ x) ,
EFFECT: On(b, Table) ∧ Clear(x) ∧ ¬ On(b, x)

รูปที่ 8.3 การบรรยายปัญหา Blocks world

การวางแผนแก้ปัญหาวางกล่อง 3 กล่องซ้อนกันเป็นตั้ง คำตอบหนึ่งได้แก่
[Move(B, Table, C), Move(A, Table, B)]

8.3 การวางแผนโดยมีอันดับบางส่วน (Partial-Order Planning)

การวางแผนที่แบ่งปัญหาออกเป็นปัญหาย่อยได้หลาย ๆ ปัญหา จะมีเป้าหมายย่อยหลายเป้าหมาย และทำงานเพื่อให้บรรลุเป้าหมายย่อยแต่ละอันโดยอิสระไม่เกี่ยวกัน เมื่อเป็นเช่นนี้ผู้วางแผนสามารถทำงานกับเรื่องที่เห็นว่าสำคัญกว่า หรือปัญหาที่มีการตัดสินใจได้ชัดเจนก่อน ไม่จำเป็นต้องทำงานไปตามอันดับของแผนก่อนหลังตามลำดับการแยกเป้าหมายย่อย

อัลกอริทึมการวางแผนที่สามารถบรรจุแอคชัน 2 แอคชันไว้ในแผนโดยไม่ต้องระบุว่าแผนใดต้องทำก่อน เรียกว่า การวางแผนที่เรียงอันดับเพียงบางส่วน (Partial-order planner) พิจารณาปัญหาตัวอย่างในการใส่รองเท้าและถุงเท้าคู่หนึ่งดังนี้

Goal(RightShoeOn ∧ LeftShoeOn)

Init( )

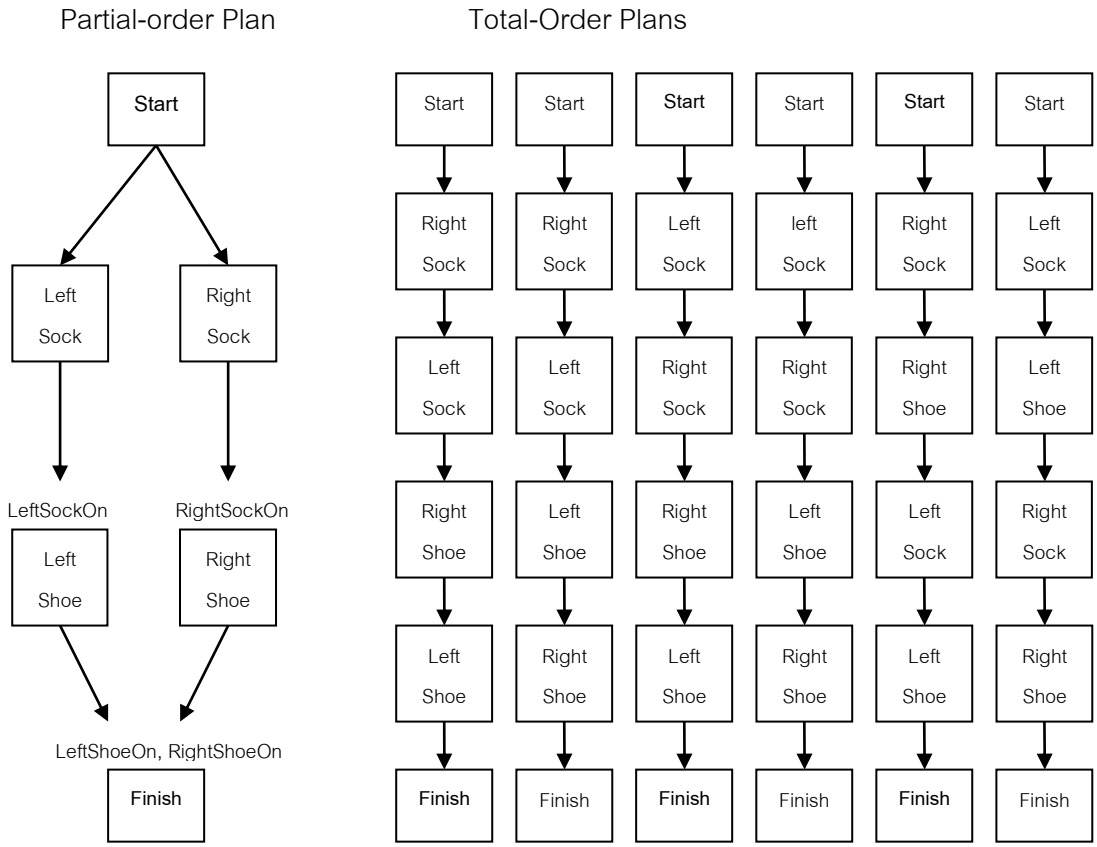
Action(RightShoe, PRECOND: RightSockOn, EFFECT: RightShoeOn)

Action(RightSock, EFFECT: RightSockOn)

Action(LeftShoe, PRECOND: LeftSockOn, EFFECT: LeftShoeOn)

Action(LeftSock, EFFECT: LeftSockOn).

ผู้วางแผนควรจะเห็นได้ว่า มี 2 แอคชันที่ต่อกันเป็นลำดับ (Sequence) คือ RightSock ตามด้วย RightShoe แล้วจึงจะบรรลุจุดประสงค์ แรกของเป้าหมาย (RightShoeOn) อีกลำดับหนึ่งคือ LeftSock ตามด้วย LeftShoe แล้วจึงจะบรรลุจุดประสงค์ที่ 2 (LeftShoeOn) เมื่อรวมผลของ 2 ลำดับที่เป็นเป้าหมายย่อยนี้เข้าด้วยกัน จะกลายเป็นคำตอบสุดท้ายของแผน เมื่อใช้คำว่าลำดับ แสดงว่าต้องเรียงอันดับการทำงาน เช่น ต้องทำ RightSock ให้เสร็จก่อน แล้วจึงทำ RightShoe ได้แต่ลำดับของแอคชันทั้งสองลำดับทำงานเป็นอิสระไม่เกี่ยวข้องกัน แอคชันของลำดับหนึ่งไม่จำเป็นต้องถูกบังคับให้ทำก่อนหรือหลังแอคชันของอีกลำดับหนึ่ง จึงเป็นแผนที่จัดอันดับเพียงบางส่วนเท่านั้น รูปที่ 8.4 แสดงแผนการทำงานซึ่งเป็นคำตอบสำหรับปัญหาการใส่รองเท้า โดยแทนคำตอบด้วยกราฟของแอคชัน มี Start และ Finish เป็นจุดเริ่มต้นและจุดจบของแผน ถือว่าทั้งสองเป็นแอคชันด้วยเพื่อให้ดูง่าย เพราะในแผน ทุกอย่างคือแอคชัน คำตอบที่หาได้คือแผนที่เรียงอันดับบางส่วนที่เป็นไปได้ทั้งหมดจำนวน 6 แผน



รูปที่ 8.4 แผนที่เรียงอันดับเพียงบางส่วน (รูปซ้าย) และแผนที่เรียงลำดับทั้งหมด (รูปขวา) สำหรับปัญหาการใส่รองเท้าและถุงเท้า

วิธีสร้างแผนแบบเรียงอันดับบางส่วนให้เกิดขึ้นจริง (ในระดับ Implement) ทำได้โดยการค้นหาไปใน Space ของแผนแบบเรียงอันดับบางส่วน (ต่อไปจะเรียกสั้น ๆ ว่าแผน) เริ่มต้นจากแผนเปล่า แล้วหาทางกลั่นกรองนำแอคชันมาใส่ลงในแผน จนกระทั่งได้แผนที่สมบูรณ์ นำมาใช้แก้ปัญหาได้ แอคชันที่ได้จากการค้นหานี้อาจจะยังไม่ใช่แอคชันจริงในโลก แต่เป็นแอคชันของแผน ซึ่งจะต้องเติมลงไปในแต่ละขั้นตอนของแผน และต้องกำหนดอันดับก่อนหลังด้วยว่าแอคชันใดต้องอยู่ก่อนแอคชันใด การสร้างแผนจึงมีลักษณะงานคล้ายกับปัญหาการค้นหา เพียงแต่สถานะของปัญหาคือแผน ในที่นี้จึงใช้คำว่าแผน แทนคำว่าสถานะ ส่วนวิธีการค้นหาจะใช้วิธีค้นหาแบบไม่มีข้อมูล หรือแบบใช้ฮิวริสติกช่วยก็ได้

แผนมีองค์ประกอบ 4 ข้อดังนี้

1. เซตของแอคชันที่ก่อให้เกิดขั้นตอนของแผน เซตนี้มาจากเซตของแอคชันในปัญหาที่ต้องการวางแผน มีแผนว่างเป็นจุดเริ่มต้น แผนว่างประกอบด้วยแอคชัน Start และ Finish เท่านั้น

Start ไม่มีเงื่อนไขก่อนหน้า แต่มีผลเป็นค่าข้อความทั้งหมดในสถานะเริ่มต้นของปัญหาที่ต้องการวางแผน ส่วน Finish มีเงื่อนไขก่อนหน้าเป็นค่าข้อความที่เป็นเป้าหมายของปัญหาที่ต้องการวางแผน และไม่มีส่วนผล

2. เซตของข้อจำกัดของการเรียงอันดับ (Ordering constraints) ข้อจำกัดของการเรียงอันดับอยู่ในรูป A $\prec$ B อ่านว่า “A ก่อน B” หมายความว่าต้องทำแอคชัน A ก่อนแอคชัน B แต่ไม่จำเป็นต้องทำก่อนหน้าในทันที หรือทำติดต่อกัน ข้อจำกัดการเรียงอันดับจะต้องบอกถึงการเรียงอันดับแบบบางส่วนได้อย่างถูกต้อง การเกิดวงรอบ (cycle) เช่น A $\prec$ B และ B $\prec$ A ถือว่ามีการขัดแย้งเกิดขึ้น ข้อจำกัดการเรียงอันดับที่ทำให้เกิดวงรอบเช่นนี้ไม่สามารถนำมาใส่ในแผนได้

3. เซตของ Causal link การ เชื่อมระหว่างแอคชัน A และ B โดยเขียนลูกศรเชื่อมเรียกว่ามี Causal link ภายในแผน เขียนว่า A $\xrightarrow{p}$ B อ่านว่า A บรรลุผล p เพื่อ B ตัวอย่างเช่น

RightSock $\xrightarrow{\text{RighSockOn}}$ RightShoe

เป็นการกล่าวว่า RightSockOn เป็นผลของแอคชัน RightSock และเป็นเงื่อนไขก่อนหน้าของ RightShoe และยังบอกได้ว่า RightSockOn ต้องเป็นจริงนับตั้งแต่ช่วงเวลาการทำ RightSock จนถึง RightShoe

เมื่อเป็นเช่นนี้ หากมีแอคชันใหม่ C อีกแอคชันหนึ่ง แผนไม่สามารถเติมแอคชันใหม่ C ลงไปได้ถ้าแอคชัน C มีความขัดแย้งกับ Causal link เรากล่าวว่า แอคชัน C จะขัดแย้งกับ A $\xrightarrow{p}$ B ถ้า C ก่อให้เกิดผล ¬p และถ้า C สามารถทำได้หลัง A และทำได้ก่อน B บางคนอาจจะเรียก Causal link ว่า Protection interval เพราะลิงก์ A $\xrightarrow{p}$ B กันไม่ให้ p เปลี่ยนเป็นนิเสธในระหว่างช่วงการทำงานจาก A ไป B

4. เซตของเงื่อนไขก่อนหน้าแบบเปิด (Open precondition) เงื่อนไขก่อนหน้าแบบเปิดหมายถึงเงื่อนไขที่ไม่มีแอคชันใดทำให้บรรลุได้ ผู้วางแผนต้องพยายามทำให้เซตนี้เป็นเซตว่าง

ตัวอย่างแผนจากปัญหาใส่รองเท้าถุงเท้าในรูปที่ 8.4 มีองค์ประกอบดังนี้

Actions: { RightSock, RightShoe, LeftSock, LeftShoe, Start, Finish }

Orderings: { RightSock $\prec$ RightShoe, LeftSock $\prec$ LeftShoe }

Links: { RightSock $\xrightarrow{\text{RghtSockOn}}$ RightShoe, LeftSock $\xrightarrow{\text{LeftSockOn}}$ LeftShoe,
RightShoe $\xrightarrow{\text{RightShoeOn}}$ Finish, LeftShoe $\xrightarrow{\text{LeftShoeOn}}$ Finish }

Open Preconditions: { }

แผนที่ไม่มีวงรอบ ไม่ขัดแย้งกับ Causal link และไม่มีเงื่อนไขก่อนหน้าแบบเปิด จะเป็นคำตอบ (Solution) ของปัญหา

ต่อไปจะดำเนินการตามวิธีค้นหาเพื่อกำหนดสร้างตัวแผนขึ้น ซึ่งก็เหมือนกับการสร้างตัวปัญหา (Problem formulation) ทั่วไป แต่ปัญหาการวางแผนจะแสดงในรูปตรรกศาสตร์ประพจน์ และโดยนิยามของปัญหา จะต้องมีแผน (สถานะ) เริ่มต้น แอคชัน และการทดสอบเป้าหมาย

1. แผนเริ่มต้น ประกอบด้วย Start และ Finish มีข้อจำกัดการเรียงลำดับคือ Start $\prec$ Finish ไม่มี Causal link และขณะนี้เงื่อนไขก่อนหน้าของ Finish ทุกอันเป็นแบบเปิด

2. Successor function เกิดจากการเลือกเงื่อนไขก่อนหน้าแบบเปิด p มาเงื่อนไขหนึ่งซึ่งเป็นของแอคชัน B และสร้างแผนลูก (Successor plan) ซึ่งหมายถึงแผนการเลือกแอคชัน A ที่ทำให้เกิด p นั้น ในขั้นตอนนี้ มีวิธีป้องกันไม่ให้เกิดวงรอบ และไม่ให้เกิดการขัดแย้งกับ Causal link ดังนี้

1) เพิ่ม Causal link $A \xrightarrow{p} B$ และข้อจำกัดการเรียงอันดับ $A \prec B$ ลงไปในแผน แอคชัน A อาจจะเป็นแอคชันที่มีอยู่แล้วในแผน หรืออาจจะเป็นแอคชันใหม่ก็ได้ ถ้าเป็นแอคชั่นใหม่ ให้เพิ่มลงไปในแผน และเพิ่ม Start $\prec$ A และ A $\prec$ Finish

2) สลายความขัดแย้งระหว่าง Causal link ใหม่ กับแอคชันที่มีอยู่เดิมทั้งหมด และความขัดแย้งระหว่างแอคชัน A (ถ้าเป็นแอคชันใหม่) กับ Causal link ที่มีอยู่เดิมทั้งหมด การสลายความขัดแย้งระหว่าง $A \xrightarrow{p} B$ กับ C ทำได้โดยบังคับให้ C เกิดขึ้นก่อน A หรือให้ B เกิดก่อน C

3. การทดสอบเป้าหมาย คือการตรวจสอบว่าแผนนั้นเป็นคำตอบของปัญหาการวางแผนจริง เนื่องจากแผนที่สร้างขึ้นไม่มีวงรอบ ไม่มีข้อขัดแย้งกับ Causal link อยู่แล้ว จึงเหลือแต่ตรวจดูว่าต้องไม่มีเงื่อนไขก่อนหน้าแบบเปิด

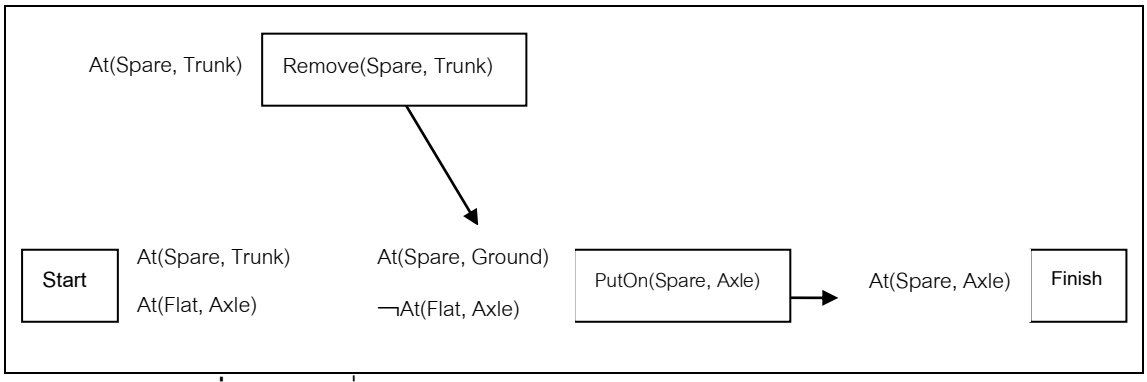
8.4 กรณีศึกษา : การวางแผนในปัญหายางแบน

ปัญหายางแบนได้กล่าวไว้ในหัวข้อ 8.2.3 และบรรยายปัญหาไว้ในรูปที่ 8.2 การค้นหาคำตอบเริ่มจากแผนเริ่มต้น ประกอบด้วย Start ซึ่งมีผลเป็น At(Spare, Trunk) $\wedge$ At(Flat, Axle) และ Finish ซึ่งมีเงื่อนไขก่อนหน้าเป็น At(Spare, Axle) เราต้องสร้างแผนลูกโดยเลือกหยิบเงื่อนไขก่อนหน้าแบบเปิดมาอันหนึ่ง แล้วเลือกหาแอคชันที่สามารถกระทำเงื่อนไขนี้ให้บรรลุผลได้ โดยในที่นี้จะเลือกแบบสุ่มไม่ต้องอาศัยฮิวริสติก มีลำดับการเหตุการณ์ที่เกิดขึ้นดังนี้

1. เลือก At(Spare, Axle) ซึ่งเป็นเงื่อนไขก่อนหน้าแบบเปิดที่มีเพียงเงื่อนไขเดียวขณะนี้ของ Finish แล้วเลือกแอคชันที่ทำสิ่งนี้ได้สำเร็จ (ในขณะนี้มีเพียงแอคชันเดียว) คือ PutOn(Spare, Axle)

2. เลือก At(Spare, Ground) ซึ่งเป็นเงื่อนไขก่อนหน้าของ PutOn(Spare, Axle) แล้วเลือกแอคชันที่ทำสิ่งนี้ได้สำเร็จ (ในขณะนี้มีเพียงแอคชันเดียว) คือ Remove(Spare, Trunk)

ดูรูปที่ 8.5 ประกอบเหตุการณ์ที่ดำเนินมาถึงจุดนี้

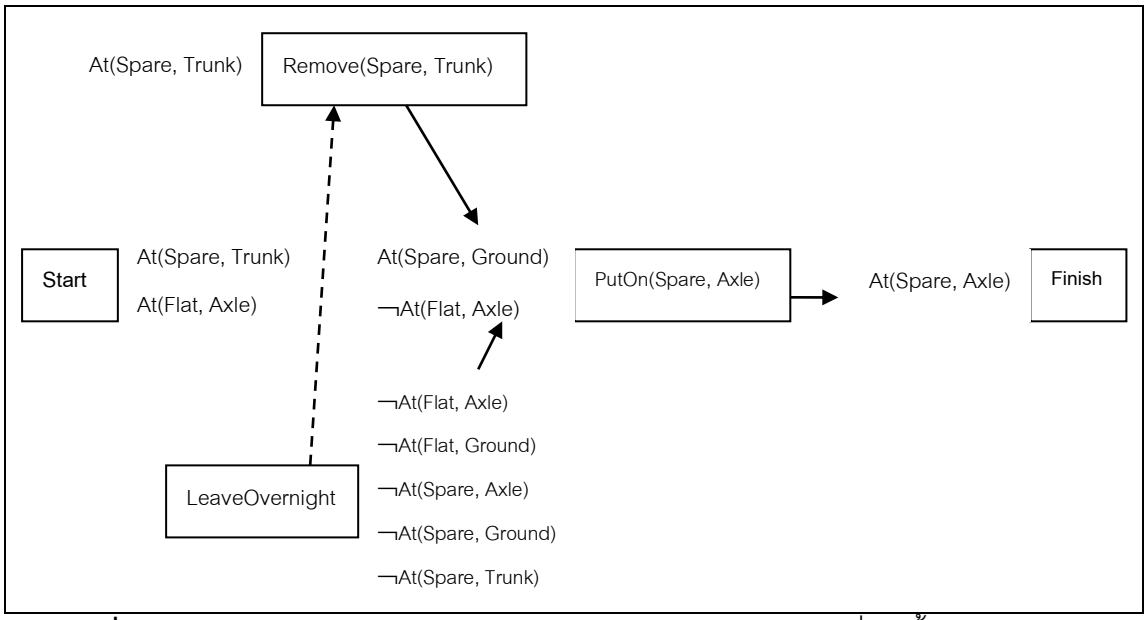


รูปที่ 8.5 แผนซึ่งเรียงอันดับบางส่วนในปัญหายางแบน (ยังไม่สมบูรณ์)

3. เลือก ¬At(Flat, Axle) ซึ่งเป็นเงื่อนไขก่อนหน้าของ PutOn(Spare, Axle) เพื่อให้มีความแตกต่าง จะเลือก LeaveOvernight แทนที่จะเลือก Remove(Flat, Axle) สังเกตดูว่า LeaveOvernight ก็มีผลเป็น ¬ At(Spare, Ground) เช่นกัน ซึ่งหมายความว่าเกิดการขัดแย้งกับ Causal link

Remove(Spare, Trunk) $\xrightarrow{\text{At(Space, Ground)}}$ PutOn(Spare, Axle)

การสลายความขัดแย้งนี้ต้องเติมข้อจำกัดของการจัดอันดับ นั่นคือกำหนดให้ LeaveOvernight เกิดขึ้นก่อน Remove(Spare, Trunk) เห็นได้จากลูกศรเส้นประตามรูปที่ 8.6



รูปที่ 8.6 แผนหลังจากเลือก LeaveOvernight และสลายการขัดแย้งที่เกิดขึ้นตามเส้นประ

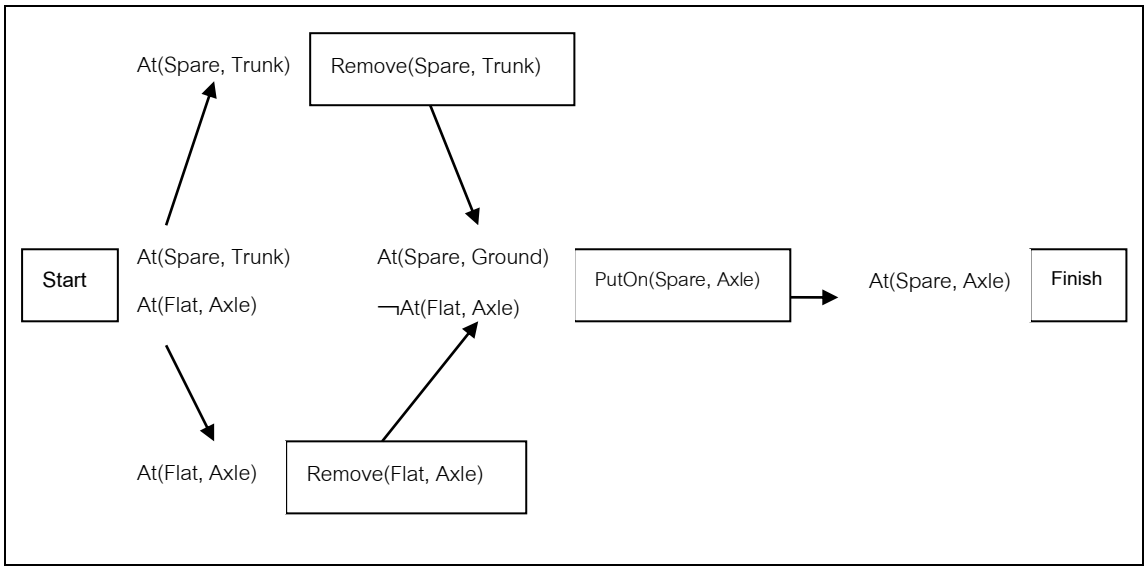
4. ยังเหลือเงื่อนไขก่อนหน้าแบบเปิดอีกเงื่อนไขหนึ่งคือ At(Spare, Trunk) ซึ่งเป็นเงื่อนไขก่อนหน้าของ Remove(Spare, Trunk) แอคชันเดียวที่สามารถบรรลุผลข้อนี้ได้คือ Start แต่ Causal link จาก Start ไป Remove(Spare, Trunk) มีความขัดแย้งกับ ¬ At(Spare, Trunk) ซึ่งเป็นผลของ LeaveOvernight แต่ครั้งนี้ไม่สามารถย้าย LeaveOvernight ไปไว้ก่อน Start ได้ เพราะ Start เป็นจุดเริ่มต้นของแผน และก็ไม่สามารถย้าย LeaveOvernight ไปไว้หลัง Remove(Spare, Trunk) ได้เช่นกัน เพราะมีข้อจำกัดการเรียงอันดับให้มันต้องอยู่ก่อน Remove(Spare, Trunk) ณ จุดนี้จึงต้องถอยหลังกลับ นำ Remove(Spare, Trunk) และ Causal link ทั้ง 2 อันออกไปก่อน แล้วกลับไปยังสถานะตามรูปที่ 8.5

เหตุการณ์นี้เป็นเครื่องพิสูจน์ว่าการทิ้งรถค้างคืนเป็นสิ่งที่ไม่ควรกระทำในปัญหายางแบน

5. กลับมาพิจารณา ¬ At(Flat, Axle) ซึ่งเป็นเงื่อนไขก่อนหน้าของ PutOn(Spare, Axle) ครั้งนี้เลือกแอคชัน Remove(Flat, Axle)

6. เลือก At(Spare, Trunk) ซึ่งเป็นเงื่อนไขก่อนหน้าของ Remove(Flat, Axle) และเลือกแอคชัน Start เพื่อให้บรรลุเป้าหมายนี้ ครั้งนี้ไม่มีความขัดแย้งเกิดขึ้น

7. เลือก At(Flat, Axle) ซึ่งเป็นเงื่อนไขก่อนหน้าของ Remove(Flat, Axle) และเลือกแอคชัน Start เพื่อให้บรรลุเป้าหมายนี้ ขณะนี้จะได้แผนที่สมบูรณ์ และไม่มีความขัดแย้งใด ๆ ตามรูปที่ 8.7



รูปที่ 8.7 คำตอบสุดท้ายของการวางแผนแก้ปัญหายางแบน

จากแผนที่สร้างขึ้นมานี้ แอคชัน Remove(Spare, Trunk) และ Remove(Flat, Axle) สามารถทำแอคชันใดก่อนหลังก็ได้ แต่ทั้ง 2 แอคชันต้องทำเสร็จก่อนที่จะทำ PutOn(Spare, Axle)

แม้ว่าปัญหายางแบนจะเป็นปัญหาวางแผนที่ง่ายมาก แต่ก็ได้แสดงถึงลักษณะเด่นของแผนที่มีการเรียงอันดับเพียงบางส่วน จะเห็นว่าการใช้ Causal link มีประโยชน์ ทำให้ตัดส่วนของการค้นหาใน Search space ได้มาก เพราะถ้าเกิดความขัดแย้งขึ้นแล้ว จะหาคำตอบไม่ได้เลย ส่วนคำตอบที่เห็นได้จากรูปที่ 8.7 นี้แม้ว่าจะเป็นเพียงแผนที่มีให้เลือกเพียง 2 แผนเท่านั้น (เมื่อทำเป็น Total-order plan) แต่ทำให้เห็นความอ่อนตัวของการวางแผน และนำมาใช้ประโยชน์ในสถานการณ์ต่าง ๆ กันได้ เช่นเมื่อต้องการเปลี่ยนยางในช่วงจราจรเร่งด่วน กับการเปลี่ยนยางในช่วงเวลาอื่น ก็ใช้คนละแผนกันได้ เป็นต้น

แบบฝึกหัดบทที่ 8

1. การทำปัญหาให้เป็นปัญหาย่อย (Decomposition) มีประโยชน์อย่างไรต่อเอเยนต์วางแผน และมีอุปสรรคหรือไม่อย่างไร
2. จากการบรรยายปัญหาการขนส่งสิ่งของทางอากาศในรูปที่ 8.1 ถ้าให้สถานการณ์ต่อไปนี้

$$At(P_1, BKK) \wedge At(P_2, CHM) \wedge Plane(P_1) \wedge Plane(P_2)$$
$$\wedge Airport(BKK) \wedge Airport(CHM)$$

จงหาว่า Fly(p, from, to) จะมีค่าเป็นอะไรได้บ้าง

3. “ปัญหาลิงกับกล้วย”

ลิงตัวหนึ่งอยู่ในห้อง มีกล้วยแขวนจากเพดานสูงเกินลิงเอื้อมถึง มีกล่องซึ่งถ้าลิงปีนกล่องนี้แล้วจะเอื้อมมือถึงกล้วยได้ เมื่อเริ่มต้น ลิงอยู่ที่จุด A กล้วยอยู่ที่จุด B และกล่องอยู่ที่จุด C ลิงและกล่องมีความสูงเท่ากับ Low แต่ถ้าลิงปีนขึ้นไปบนกล่อง จะสูงเท่ากับ High และเท่ากับระดับความสูงของกล้วย แอคชันที่ลิงทำได้มีดังนี้

Go คือ เดินจากที่หนึ่งไปอีกที่หนึ่ง

Push คือ ผลักสิ่งของจากที่หนึ่งไปอีกที่หนึ่ง

ClimbUp คือ ปีนขึ้นไปบนสิ่งของ

ClimbDown คือ ปีนลงจากสิ่งของ

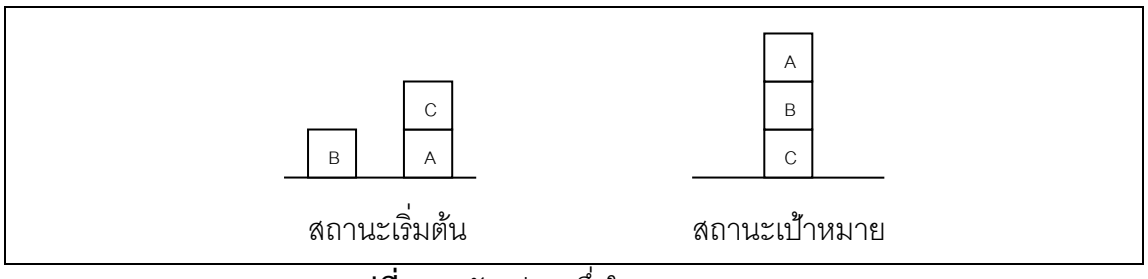
Grasp คือ จับสิ่งของ จะมีผลให้ถือสิ่งของนั้นได้ ถ้าลิงและสิ่งของอยู่ที่จุดเดียวกันและมีความสูงเท่ากัน

Ungrasp คือ ปล่อยสิ่งของ

3.1 จงเขียนบรรยายสถานะเริ่มต้น

3.2 จงเขียนบรรยายแอคชันทั้ง 6 ด้วยภาษา STRIPS

4. ปัญหา Blocks world กำหนดให้มีสถานะเริ่มต้น และสถานะเป้าหมายดังรูปที่ 8.8 จงเขียนบรรยายปัญหานี้ด้วยภาษา STRIPS แล้ววางแผนแก้ปัญหาแบบเรียงอันดับบางส่วน



รูปที่ 8.8 ตัวอย่างหนึ่งใน Blocks world