บทที่ 5
การทดสอบโปรแกรม

ในบทนี้ จะกล่าวถึงวิธีการทดสอบโปรแกรมที่ได้พัฒนาขึ้นมา ตั้งแต่การติดตั้งโปรแกรมจนถึงการทดสอบโปรแกรมด้วยกรณีศึกษา และสรุปผลการทดสอบดังนี้

5.1 ขั้นตอนการติดตั้งอาร์พีเอ็นทูคาเฟโอบีเจ

ทำการติดตั้งเครื่องมือโดยการเรียก SETUP.EXE ที่พัฒนาไว้ แล้วปฏิบัติตามขั้นตอนที่โปรแกรมแนะนำจนกว่าการติดตั้งเครื่องมือจะเสร็จเรียบร้อย สำหรับวิธีการใช้งานโปรแกรมอาร์พีเอ็นทูคาเฟโอบีเจสามารถศึกษาได้จากภาคผนวก ง

5.2 กรณีศึกษาสำหรับทดสอบโปรแกรม

ขั้นตอนการทดสอบโปรแกรม มีลำดับดังนี้

1) ตั้งกรณีศึกษาที่ใช้ในการทดสอบโปรแกรม โดยใช้ทั้งหมด 3 กรณี คือ การแปลงจากนิพจน์อินฟิกไปเป็นนิพจน์โพสต์ฟิก การแปลงจากนิพจน์โพสต์ฟิกไปเป็นนิพจน์อินฟิก และการเรียงลำดับแบบแทรก
2) สร้างแผนภาพเครือข่ายอนุภาคความต้องการของแต่ละกรณี
3) สร้างข้อมูลนำเข้าอาร์พีเอ็นเท็กซ์จากเครือข่ายอนุภาคความต้องการ
4) สร้างข้อกำหนดคาเฟโอบีเจของแต่ละกรณีศึกษา จากข้อมูลนำเข้าอาร์พีเอ็นเท็กซ์
5) ตรวจสอบความถูกต้องของวากยสัมพันธ์ของข้อกำหนดที่ได้ โดยใช้เครื่องมือแปลภาษาคาเฟโอบีเจ
6) ตรวจสอบความถูกต้องของการทำงานของข้อกำหนดที่ได้โดยใช้เครื่องมือแปลภาษาคาเฟโอบีเจ

รายละเอียดในการทดสอบทั้ง 3 กรณีศึกษามีดังนี้

5.2.1 กรณีศึกษาที่ 1 การแปลงจากนิพจน์แบบอินฟิกไปเป็นนิพจน์แบบโพสต์ฟิก (Infix to postfix conversion)

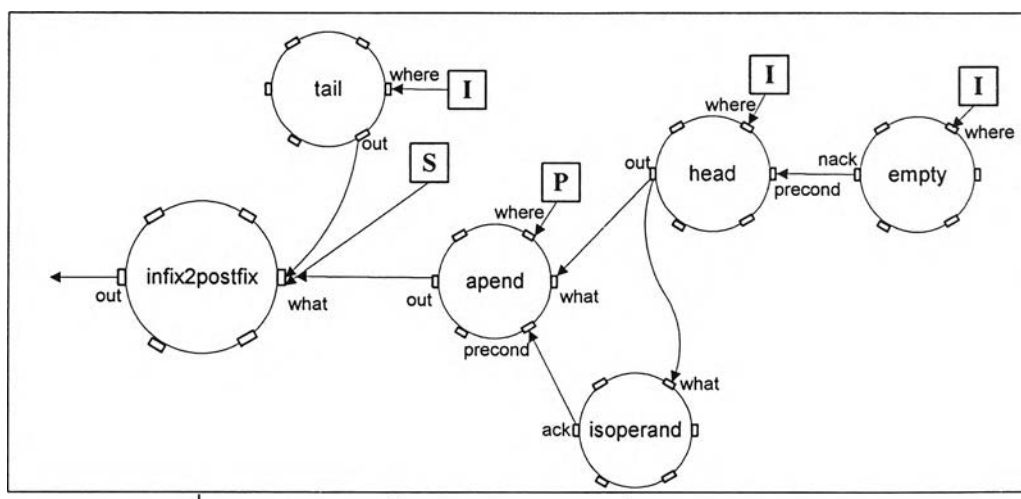
การแปลงนิพจน์อินฟิกให้เป็นโพสต์ฟิก เป็นกระบวนการที่สำคัญต่อการออกแบบโปรแกรมแปลภาษามาก [13] การแปลงนิพจน์อินฟิกให้เป็นโพสต์ฟิกต้องใช้กลไกกองซ้อนเป็นตัวช่วย นอกจากนี้แล้วต้องนิยามความสำคัญหรือลำดับก่อนหลังของตัวดำเนินการคณิตศาสตร์ สำหรับกรณีศึกษานี้จะพิจารณาตัวดำเนินการ 4 ตัว คือ + (บวก) , - (ลบ) , * (คูณ) และ / (หาร) นิพจน์ที่เขียนแบบปกติเรียกว่านิพจน์อินฟิก ทั้งนี้เนื่องจากเครื่องหมายตัวดำเนินการทาง

คณิตศาสตร์อยู่ระหว่างตัวสัญลักษณ์แทนค่าตัวเลข ส่วนนิพจน์ที่เรียกว่า นิพจน์โพสต์ฟิก เนื่องจากสัญลักษณ์ทางคณิตศาสตร์อยู่หลังตัวกระทำการ

การแปลงจากอินฟิกมาเป็นโพสต์ฟิก เมื่อพิจารณาตามเงื่อนไขที่ใช้ในการแปลงทีละขั้น โดยให้ข้อมูลเข้าคือข้อมูลที่มาจากนิพจน์อินฟิกซึ่งอยู่ในรูปของโครงสร้างข้อมูลแบบรายการ ข้อมูลผลลัพธ์คือข้อมูลซึ่งอยู่ในรูปของโครงสร้างข้อมูลแบบรายการนิพจน์โพสต์ฟิก และให้โครงสร้างข้อมูลแบบกองซ้อนเป็นที่พักข้อมูลชั่วคราว สามารถสรุปออกมาได้ทั้งหมด 9 กรณี และเมื่อสร้างเป็นแผนภาพเครือข่ายอนุภาคความต้องการได้ดังนี้

กรณีที่ 1 ถ้าข้อมูลเข้าเป็นตัวถูกดำเนินการ ให้ทำการเพิ่มข้อมูลเข้าลงไปยังรายการข้อมูลผลลัพธ์

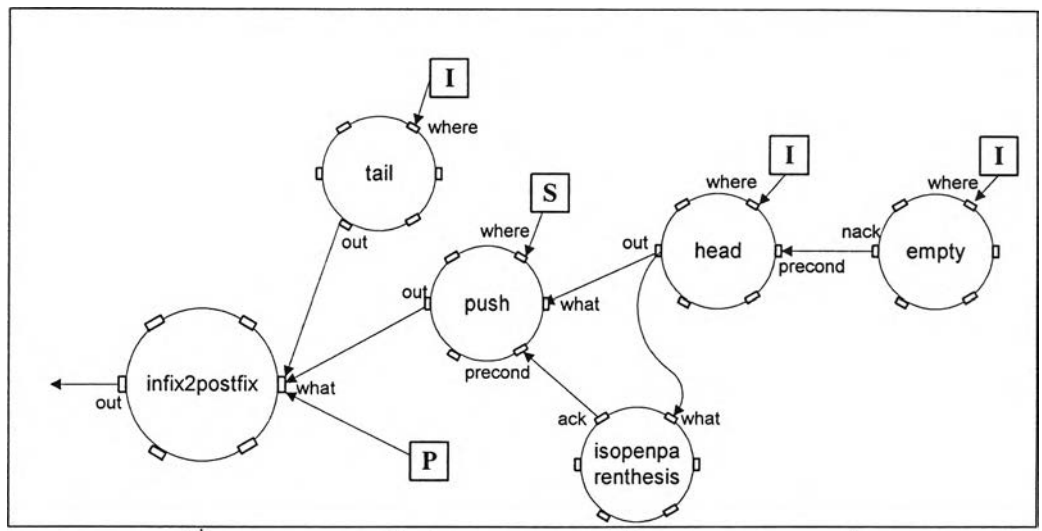
จากรูปที่ 5.1 ซึ่งเป็นแผนภาพเครือข่ายอนุภาคความต้องการที่แสดงการแปลงนิพจน์อินฟิกเป็นโพสต์ฟิกกรณีที่ 1 เมื่อทำการตรวจสอบข้อมูลของโครงสร้างแบบรายการ I (โหนด empty) ว่ามีข้อมูลอยู่จริงแล้ว จึงทำการอ่านส่วนหัวของโครงสร้างข้อมูลแบบรายการ I (โหนด head) แล้วส่งไปตรวจสอบความเป็นตัวถูกดำเนินการ (โหนด isoperand) อีกต่อหนึ่ง ถ้าการตรวจสอบนี้เป็นจริง จะนำส่วนหัวของโครงสร้างข้อมูลแบบรายการ I มาเพิ่ม (โหนด append) เข้าไปยังโครงสร้างข้อมูลแบบรายการ P ดังนั้นข้อมูลที่อยู่ที่โครงสร้างข้อมูลแบบรายการ I จะเหลือแต่เพียงส่วนหาง (โหนด tail) เท่านั้น



รูปที่ 5.1 แผนภาพเครือข่ายอนุภาคความต้องการแสดงการแปลงนิพจน์อินฟิกเป็นโพสต์ฟิกกรณีที่ 1

กรณีที่ 2 ถ้าข้อมูลเข้าเป็นเครื่องหมายวงเล็บเปิด ให้จัดเก็บไว้ที่กองซ้อน

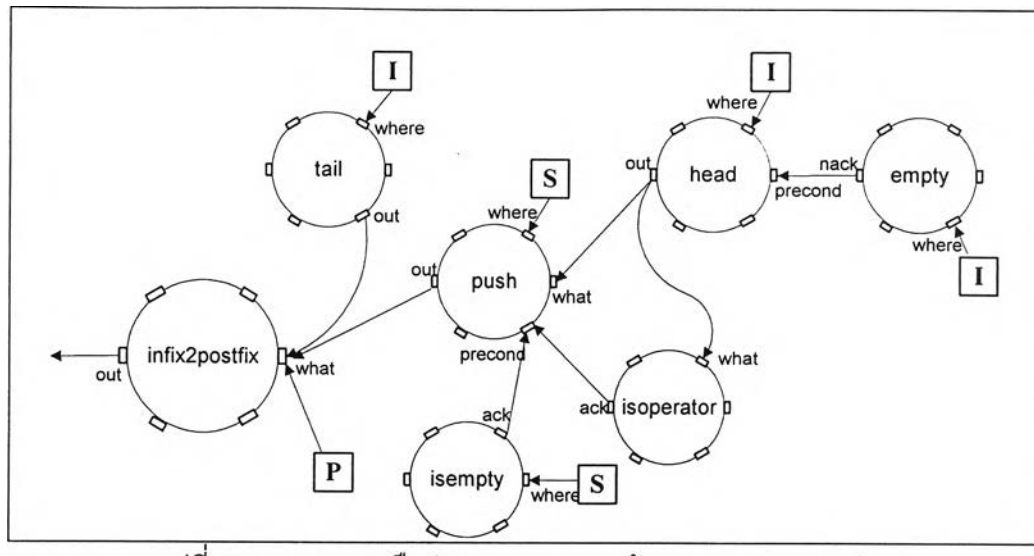
จากรูปที่ 5.2 ซึ่งเป็นแผนภาพเครือข่ายอนุภาคความต้องการที่แสดงการแปลงนิพจน์อินฟิกเป็นโพสต์ฟิกกรณีที่ 2 เมื่อทำการตรวจสอบข้อมูลของโครงสร้างแบบรายการ I (โหนด empty) ว่ามีข้อมูลอยู่จริงแล้ว จึงทำการอ่านส่วนหัวของโครงสร้างข้อมูลแบบรายการ I (โหนด head) แล้วส่งไปตรวจสอบว่าเป็นวงเล็บเปิดหรือไม่ (โหนด isopenparenthesis) อีกต่อหนึ่ง ถ้าการตรวจสอบนี้เป็นจริง จะนำส่วนหัวของโครงสร้างข้อมูลแบบรายการ I มาใส่ (โหนด push) เข้าไปยังโครงสร้างข้อมูลแบบกองซ้อน S ดังนั้นข้อมูลที่อยู่ที่โครงสร้างข้อมูลแบบรายการ I จะเหลือแต่เพียงส่วนหาง (โหนด tail) เท่านั้น



รูปที่ 5.2 แผนภาพเครือข่ายอนุภาคความต้องการแสดงการแปลงนิพจน์อินฟิกเป็นโพสต์ฟิกกรณีที่ 2

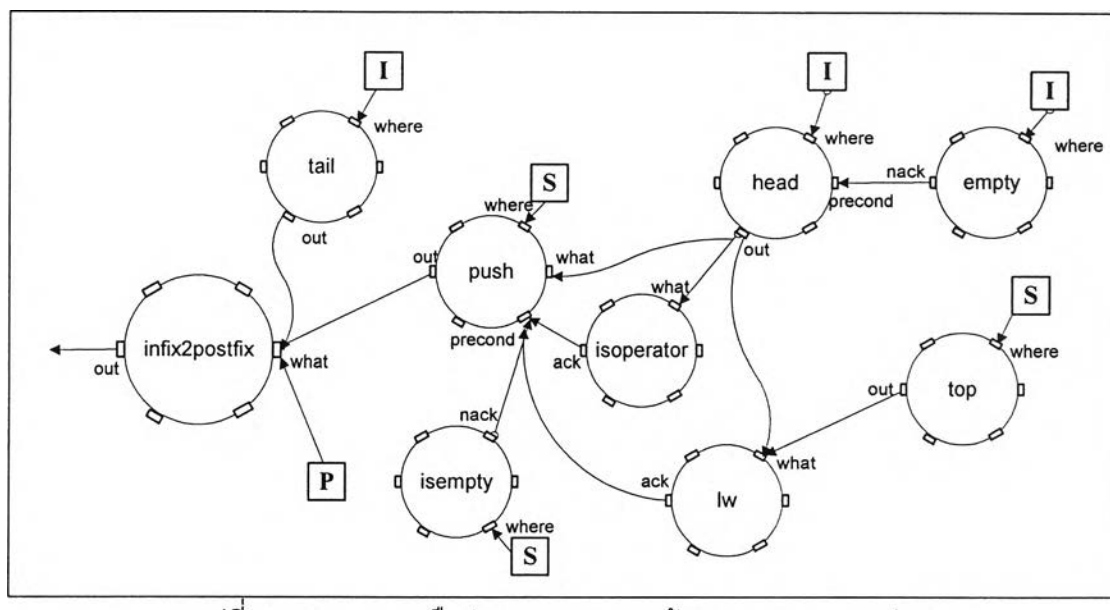
กรณีที่ 3 ถ้าข้อมูลเข้าเป็นตัวดำเนินการ ให้นำข้อมูลเข้ามาเก็บไว้ที่กองซ้อน ในกรณีที่กองซ้อนนั้นไม่มีข้อมูลอื่นอยู่

จากรูปที่ 5.3 ซึ่งเป็นแผนภาพเครือข่ายอนุภาคความต้องการที่แสดงการแปลงนิพจน์อินฟิกเป็นโพสต์ฟิกกรณีที่ 3 เมื่อทำการตรวจสอบข้อมูลของโครงสร้างแบบรายการ I (โหนด empty) ว่ามีข้อมูลอยู่จริงแล้ว จึงทำการอ่านส่วนหัวของโครงสร้างข้อมูลแบบรายการ I (โหนด head) แล้วส่งไปตรวจสอบว่าเป็นตัวดำเนินการหรือไม่ (โหนด isoperator) อีกต่อหนึ่ง ถ้าการตรวจสอบนี้เป็นจริง และการตรวจสอบว่าโครงสร้างแบบกองซ้อน S (โหนด isempty) ไม่มีข้อมูลอยู่จริง จะนำส่วนหัวของโครงสร้างข้อมูลแบบรายการ I มาใส่ (โหนด push) เข้าไปยังโครงสร้างข้อมูลแบบกองซ้อน S ดังนั้นข้อมูลที่อยู่ที่โครงสร้างข้อมูลแบบรายการ I จะเหลือแต่เพียงส่วนหาง (โหนด tail) เท่านั้น



รูปที่ 5.3 แผนภาพเครือข่ายอนุภาคความต้องการแสดงการแปลง
นิพจน์อินฟิกเป็นโพสต์ฟิกกรณีที่ 3

กรณีที่ 4 ถ้าข้อมูลเข้าเป็นตัวดำเนินการ และส่วนบนสุดของกองซ้อนมีค่าการกระทำก่อน (Precedence) ต่ำกว่าข้อมูลเข้า ให้เก็บข้อมูลเข้าไว้ที่กองซ้อน

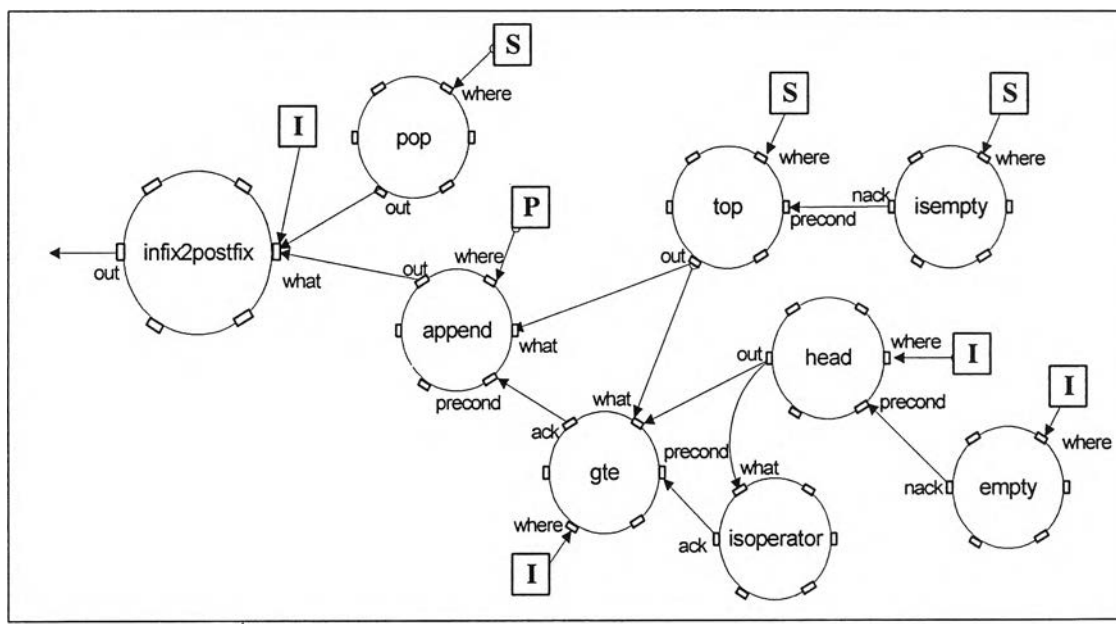


รูปที่ 5.4 แผนภาพเครือข่ายอนุภาคความต้องการแสดงการแปลง
นิพจน์อินฟิกเป็นโพสต์ฟิกกรณีที่ 4

จากรูปที่ 5.4 ซึ่งเป็นแผนภาพเครือข่ายอนุภาคความต้องการที่แสดงการแปลงนิพจน์อินฟิกเป็นโพสต์ฟิกกรณีที่ 4 เมื่อทำการตรวจสอบข้อมูลของโครงสร้างแบบรายการ I (โหนด empty) ว่ามีข้อมูลอยู่จริงแล้ว จึงทำการอ่านส่วนหัวของโครงสร้างข้อมูลแบบรายการ I (โหนด head) แล้วส่งไปตรวจสอบว่าเป็นตัวดำเนินการหรือไม่ (โหนด isoperator) อีกต่อหนึ่ง และจะทำ

การตรวจสอบว่าโครงสร้างแบบกองซ้อน S (โหนด isempty) มีข้อมูลอยู่จริงหรือไม่ จากนั้นจะนำส่วนบนสุดของโครงสร้างข้อมูลแบบกองซ้อน S (โหนด top) มาทำการเปรียบเทียบความสำคัญกับส่วนหัวของโครงสร้างข้อมูลแบบรายการ I ถ้าส่วนบนสุดของกองซ้อน S มีค่าความสำคัญน้อยกว่า (โหนด lw) จะนำส่วนหัวของโครงสร้างข้อมูลแบบรายการ I มาใส่ (โหนด push) เข้าไปยังโครงสร้างข้อมูลแบบกองซ้อน S ดังนั้นข้อมูลที่อยู่ที่โครงสร้างข้อมูลแบบรายการ I จะเหลือแต่เพียงส่วนหางเท่านั้น (โหนด tail)

กรณีที่ 5 ถ้าข้อมูลเข้าเป็นตัวดำเนินการ และส่วนบนสุดของกองซ้อนมีค่าการกระทำก่อนสูงกว่าหรือเท่ากับข้อมูลนำเข้า ให้นำค่าบนสุดของกองซ้อนมาใส่ยังรายการข้อมูลผลลัพธ์



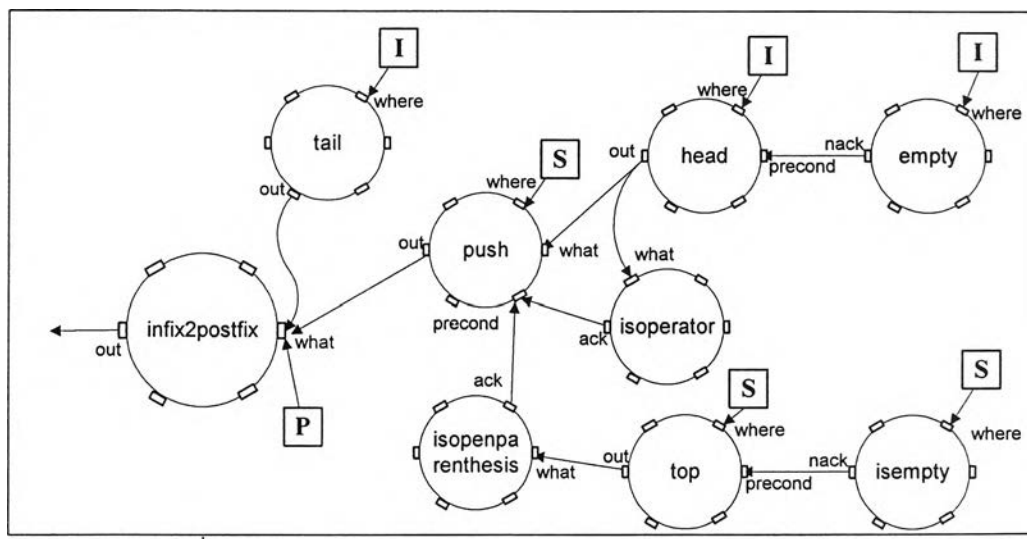
รูปที่ 5.5 แผนภาพเครือข่ายอนุภาคความต้องการแสดงการแปลงนิพจน์อินฟิกเป็นโพสต์ฟิกกรณีที่ 5

จากรูปที่ 5.5 ซึ่งเป็นแผนภาพเครือข่ายอนุภาคความต้องการที่แสดงการแปลงนิพจน์อินฟิกเป็นโพสต์ฟิกกรณีที่ 5 เมื่อทำการตรวจสอบข้อมูลของโครงสร้างแบบรายการ I (โหนด empty) ว่ามีข้อมูลอยู่จริงแล้ว จึงทำการอ่านส่วนหัวของโครงสร้างข้อมูลแบบรายการ I (โหนด head) แล้วส่งไปตรวจสอบว่าเป็นตัวดำเนินการหรือไม่ (โหนด isoperator) อีกต่อหนึ่ง และจะทำการตรวจสอบว่าโครงสร้างแบบกองซ้อน S (โหนด isempty) มีข้อมูลอยู่จริงหรือไม่ จากนั้นจะนำส่วนบนสุดของโครงสร้างข้อมูลแบบกองซ้อน S (โหนด top) มาทำการเปรียบเทียบความสำคัญกับส่วนหัวของโครงสร้างข้อมูลแบบรายการ I ถ้าส่วนบนสุดของกองซ้อน S มีค่าความสำคัญ

มากกว่าหรือเท่ากัน (โหนด gte) จะนำส่วนบนสุดของโครงสร้างข้อมูลแบบกองซ้อน S มา เพิ่ม (โหนด append) เข้าไปยังโครงสร้างข้อมูลแบบรายการ P ดังนั้นข้อมูลที่อยู่ที่โครงสร้างข้อมูลแบบกอง ซ้อน S จะถูกลดลง (โหนด pop) ไปหนึ่งค่า

กรณีที่ 6 ถ้าข้อมูลนำเข้าเป็นตัวดำเนินการ และส่วนบนสุดของกองซ้อนเป็นเครื่องหมายวงเล็บเปิด ให้เก็บข้อมูลเข้าลงไปยังกองซ้อน

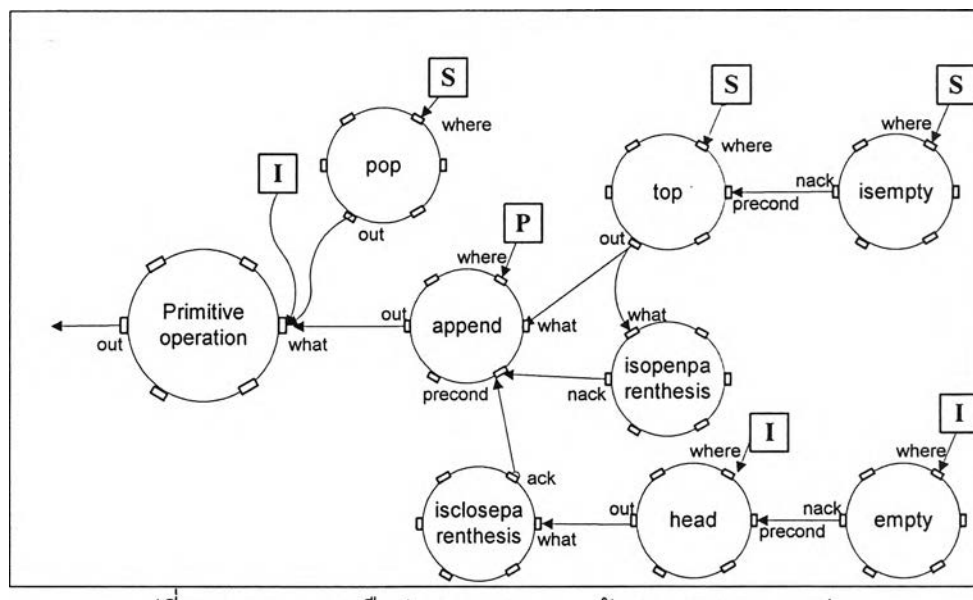
จากรูปที่ 5.6 ซึ่งเป็นแผนภาพเครือข่ายอนุภาคความต้องการที่แสดงการแปลงนิพจน์อินฟิกเป็นโพสต์ฟิกกรณีที่ 6 เมื่อทำการตรวจสอบข้อมูลของโครงสร้างแบบรายการ I (โหนด empty) ว่ามีข้อมูลอยู่จริงแล้ว จึงทำการอ่านส่วนหัวของโครงสร้างข้อมูลแบบรายการ I (โหนด head) แล้วส่งไปตรวจสอบว่าเป็นตัวดำเนินการหรือไม่ (โหนด isoperator) อีกต่อหนึ่ง และจะทำการตรวจสอบว่าโครงสร้างแบบกองซ้อน S (โหนด isempty) มีข้อมูลอยู่จริงหรือไม่ จากนั้นจะนำส่วนบนสุดของโครงสร้างข้อมูลแบบกองซ้อน S (โหนด top) มาตรวจสอบว่าเป็นวงเล็บ เปิดหรือไม่ (โหนด isopenparenthesis) ถ้าใช่ จะนำส่วนหัวของโครงสร้างข้อมูลแบบรายการ I มาใส่ (โหนด push) เข้าไปยังโครงสร้างข้อมูลแบบกองซ้อน S ดังนั้นข้อมูลที่อยู่ที่โครงสร้างข้อมูล แบบรายการ I จะเหลือแต่เพียงส่วนหาง (โหนด tail) เท่านั้น



รูปที่ 5.6 แผนภาพเครือข่ายอนุภาคความต้องการแสดงการแปลง
นิพจน์อินฟิกเป็นโพสต์ฟิกกรณีที่ 6

กรณีที่ 7 ถ้าข้อมูลเข้าเป็นเครื่องหมายวงเล็บปิด และส่วนบนสุดของกองซ้อนไม่ใช้เครื่องหมายวงเล็บเปิด ให้นำค่าบนสุดของกองซ้อนมาใส่ยังข้อมูลผลลัพธ์

จากรูปที่ 5.7 ซึ่งเป็นแผนภาพเครือข่ายอนุภาคความต้องการที่แสดงการแปลงนิพจน์อินฟิกเป็นโพสต์ฟิกกรณีที่ 7 เมื่อทำการตรวจสอบข้อมูลของโครงสร้างแบบรายการ I (โหนด empty) ว่ามีข้อมูลอยู่จริงแล้ว จึงทำการอ่านส่วนหัวของโครงสร้างข้อมูลแบบรายการ I (โหนด head) แล้วส่งไปตรวจสอบว่าเป็นวงเล็บปิดหรือไม่ (โหนด iscloseparenthesis) อีกต่อหนึ่ง และจะทำการตรวจสอบว่าโครงสร้างแบบกองซ้อน S (โหนด isempty) มีข้อมูลอยู่จริงหรือไม่ จากนั้นจะนำส่วนบนสุดของโครงสร้างข้อมูลแบบกองซ้อน S (โหนด top) มาตรวจสอบว่าเป็นวงเล็บเปิดหรือไม่ (โหนด isopenparenthesis) ถ้าไม่ใช่จะนำส่วนบนสุดของโครงสร้างข้อมูลแบบ กองซ้อน S มาเพิ่ม (โหนด append) เข้าไปยังโครงสร้างข้อมูลแบบรายการ P ดังนั้นข้อมูลที่อยู่ที่โครงสร้างข้อมูลแบบกองซ้อน S จะถูกลดลง (โหนด pop) ไปหนึ่งค่า

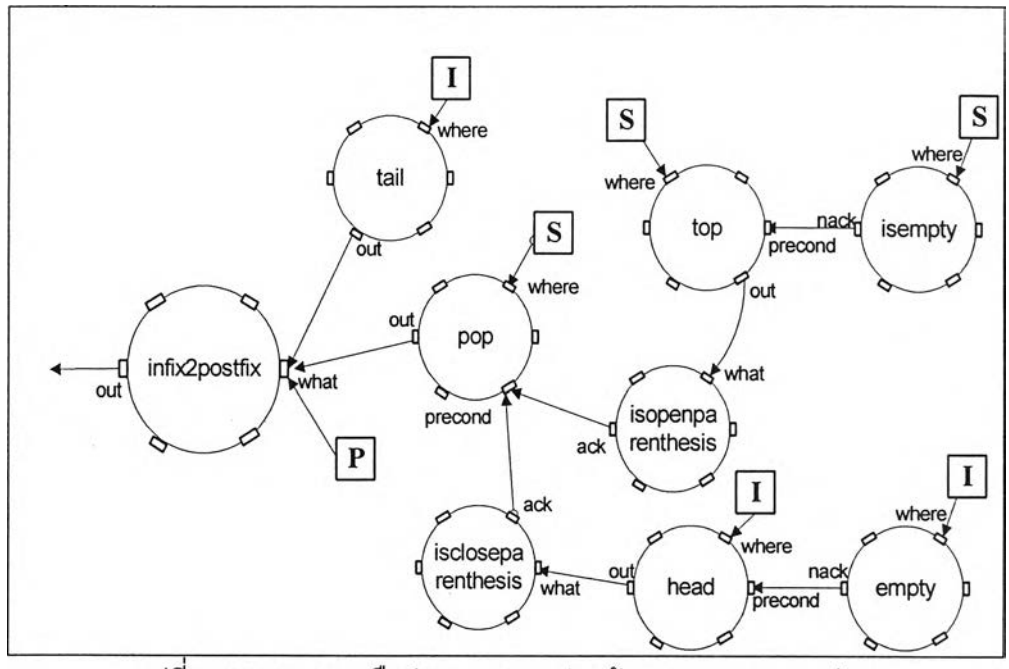


รูปที่ 5.7 แผนภาพเครือข่ายอนุภาคความต้องการแสดงการแปลง
นิพจน์อินฟิกเป็นโพสต์ฟิกกรณีที่ 7

กรณีที่ 8 ถ้าข้อมูลเข้าเป็นเครื่องหมายวงเล็บปิด และส่วนบนสุดของกองซ้อนเป็นเครื่องหมายวงเล็บเปิด ให้นำค่าส่วนบนสุดของกองซ้อนออก โดยไม่ต้องจัดเก็บไว้ที่ใด

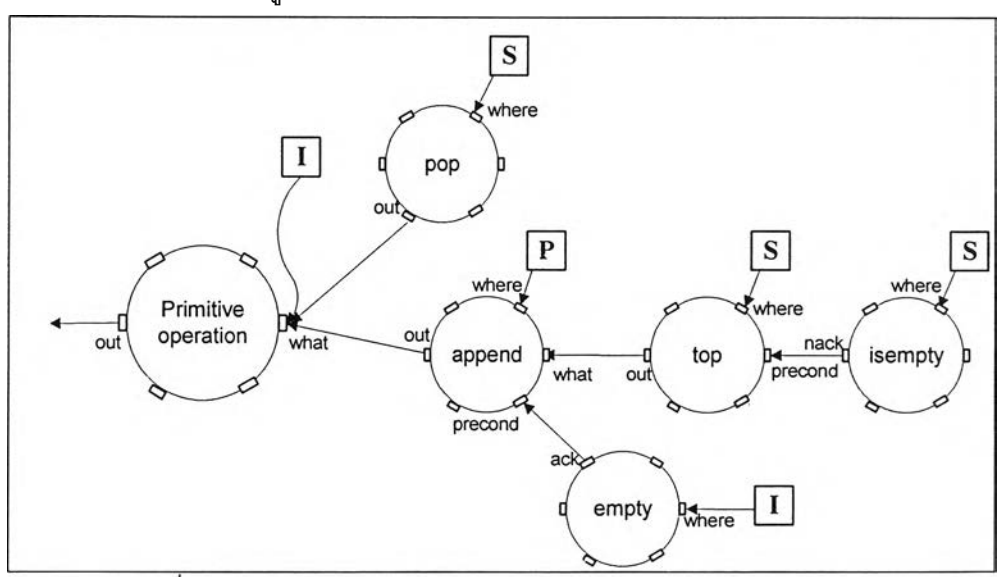
จากรูปที่ 5.8 ซึ่งเป็นแผนภาพเครือข่ายอนุภาคความต้องการที่แสดงการแปลงนิพจน์อินฟิกเป็นโพสต์ฟิกกรณีที่ 8 เมื่อทำการตรวจสอบข้อมูลของโครงสร้างแบบรายการ I (โหนด empty) ว่ามีข้อมูลอยู่จริงแล้ว จึงทำการอ่านส่วนหัวของโครงสร้างข้อมูลแบบรายการ I (โหนด head) แล้วส่งไปตรวจสอบว่าเป็นวงเล็บปิดหรือไม่ (โหนด iscloseparenthesis) อีกต่อหนึ่ง และจะทำการตรวจสอบว่าโครงสร้างแบบกองซ้อน S (โหนด isempty) มีข้อมูลอยู่จริงหรือไม่ จากนั้นจะนำส่วนบนสุดของโครงสร้างข้อมูลแบบกองซ้อน S (โหนด top) มาตรวจสอบว่าเป็นวงเล็บเปิด

หรือไม่ (โหนด isopenparenthesis) ถ้าใช่ จะทำการลดค่าของข้อมูลในโครงสร้างข้อมูลแบบกองซ้อน S (โหนด pop) ไปหนึ่งค่า และข้อมูลที่อยู่ที่โครงสร้างข้อมูลแบบรายการ I จะเหลือแต่เพียงส่วนหาง (โหนด tail) เท่านั้น



รูปที่ 5.8 แผนภาพเครือข่ายอนุภาคความต้องการแสดงการแปลงนิพจน์อินฟิกเป็นโพสต์ฟิกกรณีที่ 8

กรณีที่ 9 ถ้าไม่มีข้อมูลเข้า แต่ในกองซ้อนยังมีข้อมูลอยู่ ให้นำข้อมูลในกองซ้อนทั้งหมดมาใส่ไว้ที่รายการข้อมูลผลลัพธ์



รูปที่ 5.9 แผนภาพเครือข่ายอนุภาคความต้องการแสดงการแปลงนิพจน์อินฟิกเป็นโพสต์ฟิกกรณีที่ 9

จากรูปที่ 5.9 ซึ่งเป็นแผนภาพเครือข่ายอนุภาคความต้องการที่แสดงการแปลงนิพจน์อินฟิกเป็นโพสต์ฟิกกรณีที่ 9 เมื่อทำการตรวจสอบข้อมูลของโครงสร้างแบบรายการ I (โหนด empty) ว่าไม่มีข้อมูลแล้ว จะตรวจสอบโครงสร้างข้อมูลแบบกองซ้อน S (โหนด isempty) ว่ายังมีข้อมูลอยู่ จะทำการอ่านส่วนบนสุดของโครงสร้างข้อมูลแบบกองซ้อน S (โหนด top) มาเพิ่ม (โหนด append) เข้าไปยังโครงสร้างข้อมูลแบบรายการ P และโครงสร้างข้อมูลแบบกองซ้อน S จะถูกลดค่า (โหนด pop) ลงไปหนึ่ง

จากรูปเครือข่ายอนุภาคความต้องการที่ได้ เมื่อนำมาแปลงให้อยู่ในรูปของอาร์พีเอ็นเท็กซ์เพื่อจะทำเป็นข้อมูลนำเข้า จะได้อาร์พีเอ็นเท็กซ์ดังภาคผนวก ข และเมื่อทำการแปลงเป็นข้อกำหนดคาเฟโอบีเจ โดยใช้คลังจัดเก็บตัวดำเนินการจากภาคผนวก ค จะได้ผลลัพธ์ดังรูปที่ 5.10

การตรวจสอบความถูกต้องของมอดูลการแปลงอินฟิกเป็นโพสต์ฟิก ได้ทำการตรวจสอบโดยนำข้อกำหนดที่ได้ ใส่ไปในโปรแกรมการแปลคาเฟโอบีเจ ซึ่งผลการตรวจสอบทั้งสองส่วนมีดังนี้

- ขั้นตอนการตรวจสอบความถูกต้องตามวากยสัมพันธ์ของภาษาคาเฟโอบีเจ จะได้ผลดังรูปที่ 5.11 ซึ่งแสดงว่าข้อกำหนดของมอดูลการแปลงอินฟิกเป็นโพสต์ฟิกที่สร้างจากแผนภาพเครือข่ายอนุภาคความต้องการถูกต้องตามวากยสัมพันธ์ของภาษาคาเฟโอบีเจ
- ขั้นตอนการตรวจสอบความถูกต้องของข้อกำหนด ได้ทำการตรวจสอบความถูกต้องโดยการใส่ข้อมูลเข้าแบบอินฟิกเข้าไปและตรวจสอบผลลัพธ์ซึ่งได้ออกมาอยู่ในรูปของโพสต์ฟิกที่ถูกต้องดังรูปที่ 5.12

จากรูป 5.12 ได้ทำการตรวจสอบความถูกต้องของข้อกำหนด ได้กำหนดให้ข้อมูลทดสอบสำหรับนิพจน์อินฟิกคือ (a + b) * (c – d) และผลลัพธ์ที่ได้จากการแปลงซึ่งอยู่ในรูปของนิพจน์โพสต์ฟิก คือ ab+cd-*

5.2.2 กรณีศึกษาที่ 2 การแปลงจากนิพจน์แบบโพสต์ฟิกไปเป็นนิพจน์แบบอินฟิก (Postfix to infix conversion)

การแปลงจากโพสต์ฟิกเป็นอินฟิก จะคล้ายกับการทำอินฟิกเป็นโพสต์ฟิก หากแต่เป็นการกระทำที่กลับกัน ซึ่งขั้นตอนวิธีที่ใช้ในการแปลงจะไม่เหมือนกัน

```
mod* INFIX2POSTFIX {
pr( ISMEMBER + LIST + STACK + PRECEDENCE )
op infix2postfix : List Stack List -> List
var I : List
var P : List
var S : Stack

ceq infix2postfix(I, S, P) = infix2postfix(tail(I), S, append(P,head(I))) if isoperand(head(I))
== true and empty(I) == false .
ceq infix2postfix(I, S, P) = infix2postfix(tail(I), push(head(I), S), P) if
isopenparenthesis(head(I)) == true and empty(I) == false .
ceq infix2postfix(I, S, P) = infix2postfix(tail(I), push(head(I), S), P) if isoperator(head(I))
== true and isempty(S) == true and empty(I) == false .
ceq infix2postfix(I, S, P) = infix2postfix(tail(I), push(head(I), S), P) if isoperator(head(I))
== true and lw(top(S), head(I)) == true and isempty(S) == false and empty(I) == false .
ceq infix2postfix(I, S, P) = infix2postfix(I, pop(S), append(P, top(S))) if
isoperator(head(I)) == true and gte(top(S), head(I)) == true and isempty(S) == false and
empty(I) == false .
ceq infix2postfix(I, S, P) = infix2postfix(tail(I), push(head(I), S), P) if isoperator(head(I))
== true and isopenparenthesis(top(S)) == true and empty(I) == false and isempty(S) ==
false .
ceq infix2postfix(I, S, P) = infix2postfix(I, pop(S), append(P, top(S))) if
iscloseparenthesis(head(I)) == true and isopenparenthesis(top(S)) == false and
empty(I) == false and isempty(S) == false .
ceq infix2postfix(I, S, P) = infix2postfix(tail(I), pop(S), P)if iscloseparenthesis(head(I)) ==
true and isopenparenthesis(top(S)) == true and empty(I) == false and isempty(S) ==
false .
ceq infix2postfix(I, S, P) = infix2postfix(I, pop(S), append(P, top(S))) if empty(I) == true
and isempty(S) == false .
}
```

รูปที่ 5.10 ข้อกำหนดคาเฟโอบีเจของการแปลงนิพจน์อินฟิกเป็นโพสต์ฟิก

```
CafeOBJ> in infix2postfix
processing input : .\infix2postfix.mod
-- defining module* TOPLEVEL_*...._.....* done.
CafeOBJ>
```

รูปที่ 5.11 ผลการตรวจสอบวากยสัมพันธ์ของมอดูลการแปลงอินฟิกเป็นโพสต์ฟิก

```
CafeOBJ> open INFIX2POSTFIX
-- opening module INFIX2POSTFIX.. done.
%INFIX2POSTFIX>  ops a b : -> List .
%INFIX2POSTFIX>  op s : -> Stack .
%INFIX2POSTFIX>  eq a  = list("(", list("a", list("+", list("b", list(")", list("*", list("(", list("c",
list("-", list("d", list(")", init-list))))))))))) .
-
%INFIX2POSTFIX> eq b = init-list .
%INFIX2POSTFIX> eq s = init .
%INFIX2POSTFIX> red infix2postfix(a,s,b) .
*
-- reduce in %INFIX2POSTFIX : infix2postfix(a,s,b)
infix2postfix(init-list,init,list("a",list("b",list("+",
                                    list("c",
                                        list("d",
                                            list("-",
                                                list(
 "*",init-list))))))) : List
(0.000 sec for parse, 33196 rewrites(1.322 sec), 48177 matches)
```

นิพจน์อินฟิก
(a+b)*(c-d)

นิพจน์โพสต์ฟิก
ab+cd-*

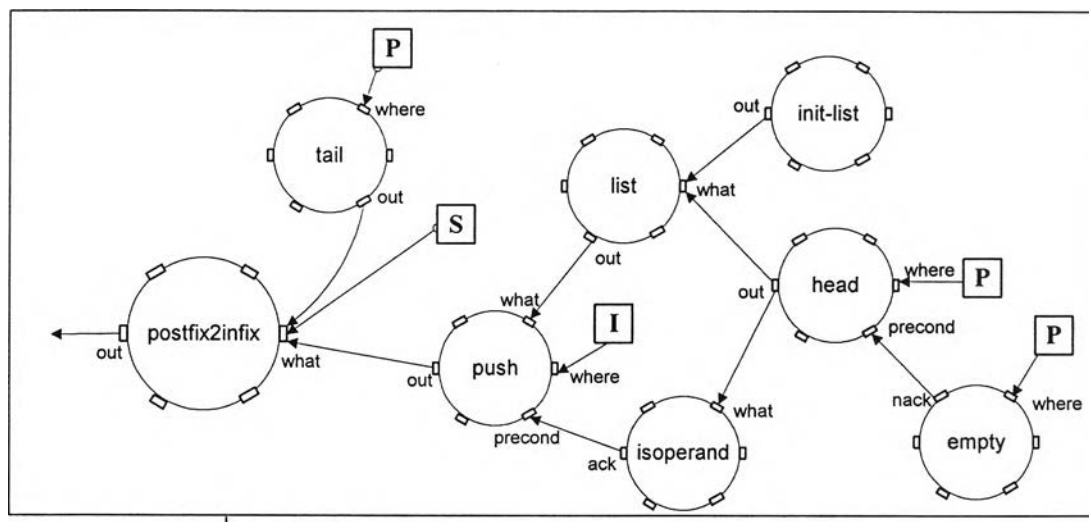
รูปที่ 5.12 ผลการตรวจสอบความถูกต้องของข้อกำหนดมอดูลการแปลงอินฟิกเป็นโพสต์ฟิก

การแปลงจากโพสต์ฟิกเป็นอินฟิก เมื่อพิจารณาเงื่อนไขในการแปลงแบบทีละขั้น โดยให้ข้อมูลเข้าคือข้อมูลที่มาจากนิพจน์โพสต์ฟิกซึ่งอยู่ในรูปของโครงสร้างข้อมูลแบบรายการ ข้อมูลผลลัพธ์คือข้อมูลซึ่งอยู่ในรูปของโครงสร้างข้อมูลแบบกองซ้อนนิพจน์อินฟิก และที่พักข้อมูลชั่ว

คราวป็นโครงสร้างข้อมูลแบบกองซ้อน สามารถสรุปเป็นเงื่อนไขในการแปลงได้ 3 กรณี และเมื่อสร้างเป็นแผนภาพเครือข่ายอนุภาคความต้องการได้ดังนี้

กรณีที่ 1 ถ้าข้อมูลเข้าเป็นตัวถูกดำเนินการ ให้นำข้อมูลเข้าที่เป็นตัวถูกดำเนินการเก็บไปยังกองซ้อนสำหรับข้อมูลผลลัพธ์

จากรูปที่ 5.13 ซึ่งเป็นแผนภาพเครือข่ายอนุภาคความต้องการที่แสดงการแปลงนิพจน์โพสต์ฟิกเป็นอินฟิกกรณีที่ 1 เมื่อทำการตรวจสอบข้อมูลของโครงสร้างแบบรายการ P (โหนด empty) ว่ามีข้อมูลอยู่จริงแล้ว จึงทำการอ่านส่วนหัวของโครงสร้างข้อมูลแบบรายการ P (โหนด head) แล้วส่งไปตรวจสอบความเป็นตัวถูกดำเนินการ (โหนด isoperand) อีกต่อหนึ่ง ถ้าการตรวจสอบนี้เป็นจริง จะนำส่วนหัวของโครงสร้างข้อมูลแบบรายการ P มาใส่ (โหนด push) เข้าไปยังโครงสร้างข้อมูลแบบกองซ้อน S ดังนั้นข้อมูลที่อยู่ที่โครงสร้างข้อมูลแบบรายการ P จะเหลือแต่เพียงส่วนหาง (โหนด tail) เท่านั้น

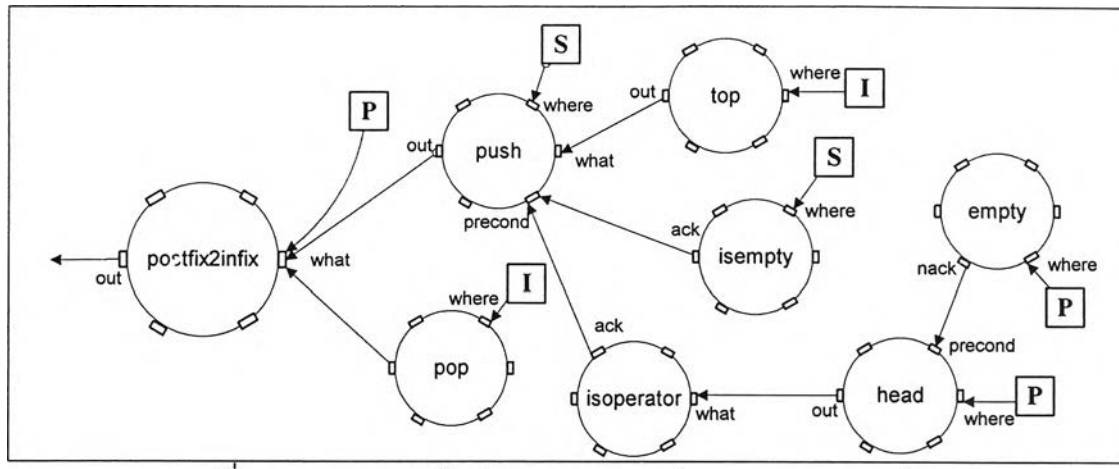


รูปที่ 5.13 แผนภาพเครือข่ายอนุภาคความต้องการแสดงการแปลงนิพจน์โพสต์ฟิกเป็นอินฟิกกรณีที่ 1

กรณีที่ 2 ถ้าข้อมูลเข้าเป็นตัวดำเนินการและกองซ้อนสำหรับพักข้อมูลว่าง ให้นำค่าบนสุดของกองซ้อนสำหรับผลลัพธ์ไปพักไว้ที่กองซ้อนสำหรับพักข้อมูลก่อน

จากรูปที่ 5.14 ซึ่งเป็นแผนภาพเครือข่ายอนุภาคความต้องการที่แสดงการแปลงนิพจน์โพสต์ฟิกเป็นอินฟิกกรณีที่ 2 เมื่อทำการตรวจสอบข้อมูลของโครงสร้างแบบรายการ P (โหนด empty) ว่ามีข้อมูลอยู่จริงแล้ว จึงทำการอ่านส่วนหัวของโครงสร้างข้อมูลแบบรายการ P (โหนด head) แล้วส่งไปตรวจสอบความเป็นตัวดำเนินการ (โหนด isoperator) อีกต่อหนึ่ง ถ้าการตรวจสอบนี้เป็นจริง จะทำการตรวจสอบโครงสร้างข้อมูลแบบกองซ้อน S (โหนด isempty) ถ้าว่าง

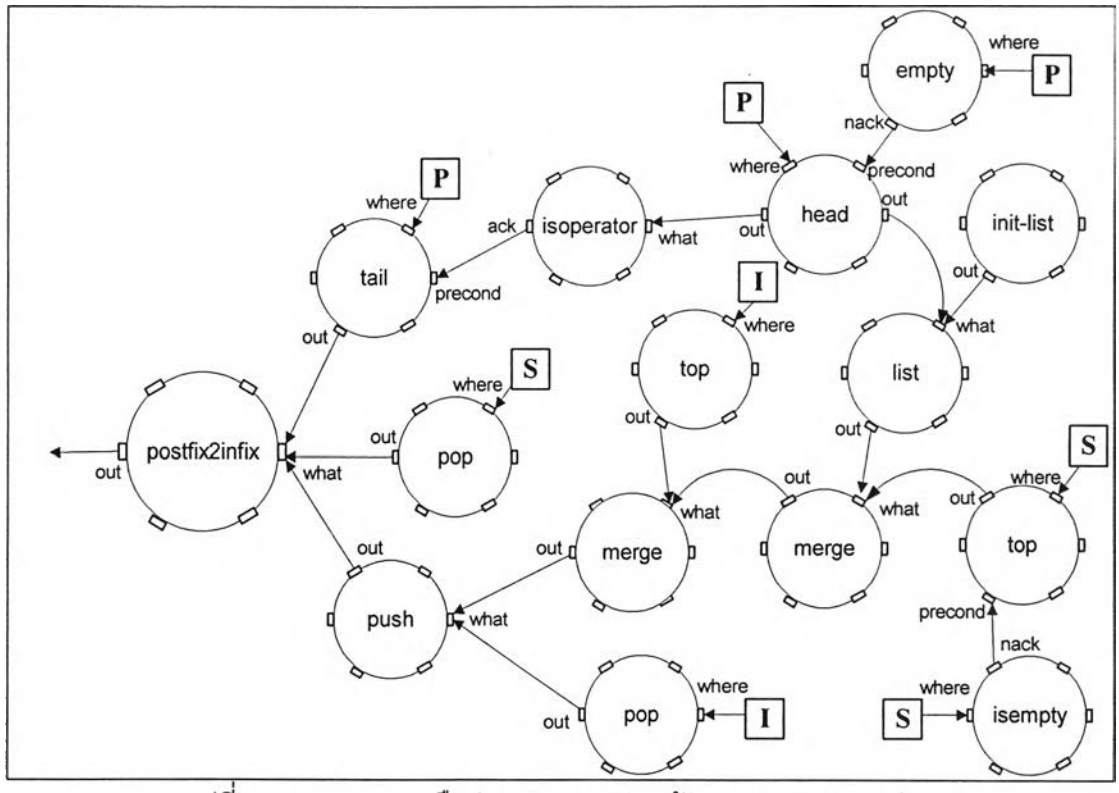
จะทำการอ่านส่วนบนสุด (โหนด top) ของโครงสร้างข้อมูลแบบกองซ้อน I มาใส่ (โหนด push) ไว้ยังโครงสร้างข้อมูลแบบกองซ้อน S จึงทำให้โครงสร้างข้อมูลแบบกองซ้อน I มีข้อมูลลดลง (โหนด pop) ไปหนึ่งค่า



รูปที่ 5.14 แผนภาพเครือข่ายอนุภาคความต้องการแสดงการแปลงนิพจน์โพสต์ฟิกเป็นอินฟิกกรณีที่ 2

กรณีที่ 3 ถ้าข้อมูลเข้าเป็นตัวดำเนินการและกองซ้อนสำหรับพักข้อมูลมีข้อมูลอยู่ ให้ทำการอ่านค่าบนสุดของกองซ้อนสำหรับผลลัพธ์ขึ้นมาใส่ตัวดำเนินการต่อท้าย และอ่านค่าบนสุดของกองซ้อนสำหรับพักข้อมูลมาใส่ต่อท้ายตัวดำเนินการอีกที จากนั้นจึงเก็บค่าทั้งหมดลงไปยังกองซ้อนสำหรับผลลัพธ์

จากรูปที่ 5.15 ซึ่งเป็นแผนภาพเครือข่ายอนุภาคความต้องการที่แสดงการแปลงนิพจน์โพสต์ฟิกเป็นอินฟิกกรณีที่ 3 เมื่อทำการตรวจสอบข้อมูลของโครงสร้างแบบรายการ P (โหนด empty) ว่ามีข้อมูลอยู่จริงแล้ว จึงทำการอ่านส่วนหัวของโครงสร้างข้อมูลแบบรายการ P (โหนด head) แล้วส่งไปตรวจสอบความเป็นตัวดำเนินการ (โหนด isoperator) อีกต่อหนึ่ง ถ้าการตรวจสอบนี้เป็นจริง จะทำการตรวจสอบโครงสร้างข้อมูลแบบกองซ้อน S (โหนด isempty) ต่อ ซึ่งถ้ามีข้อมูลเช่นเดียวกัน จะอ่านค่าบนสุดของโครงสร้างข้อมูลแบบกองซ้อน S (โหนด top) มารวมกับตัวดำเนินการที่อ่านเข้ามาก่อน (โหนด merge) โดยให้ตัวดำเนินการอยู่ข้างหน้า จากนั้นจะอ่านค่าบนสุดของโครงสร้างข้อมูลแบบกองซ้อน I (โหนด top) ขึ้น เพื่อมารวม (โหนด merge) กับตัวดำเนินการที่เพิ่งทำการรวมไป แล้วเก็บลงไปยังโครงสร้างข้อมูลแบบกองซ้อน I คืน (โหนด push) ดังนั้นข้อมูลที่อยู่ในโครงสร้างข้อมูลแบบรายการ P จึงเหลือเพียงส่วนหาง (โหนด tail) เท่านั้น และข้อมูลที่อยู่ในโครงสร้างข้อมูลแบบรายการ S จะลดค่าลงไปหนึ่งค่า (โหนด pop)



รูปที่ 5.15 แผนภาพเครือข่ายอนุภาคความต้องการแสดงการแปลงนิพจน์โพสต์ฟิกเป็นอินฟิกกรณีที่ 3

จากรูปเครือข่ายอนุภาคความต้องการที่ได้ เมื่อนำมาแปลงให้อยู่ในรูปของอาร์พีเอ็นเท็กซ์ เพื่อจะทำเป็นข้อมูลนำเข้า จะได้อาร์พีเอ็นเท็กซ์ดังภาคผนวก ข และเมื่อทำการแปลงเป็นข้อกำหนดคาเฟโอบีเจ โดยใช้คลังจัดเก็บตัวดำเนินการจากภาคผนวก ค จะได้ผลลัพธ์รูปที่ 5.16

การตรวจสอบความถูกต้องของมอดูลการแปลงโพสต์ฟิกเป็นอินฟิก ได้ทำการตรวจสอบโดยนำข้อกำหนดที่ได้ ใส่ไปในโปรแกรมการแปลคาเฟโอบีเจ ซึ่งผลการตรวจสอบทั้งสองส่วนมีดังนี้

- ขั้นตอนการตรวจสอบความถูกต้องตามวากยสัมพันธ์ของภาษาคาเฟโอบีเจ จะได้ผลดังรูปที่ 5.17 ซึ่งแสดงว่าข้อกำหนดของมอดูลการแปลงโพสต์ฟิกเป็นอินฟิกที่สร้างจากแผนภาพเครือข่ายอนุภาคความต้องการถูกต้องตามวากยสัมพันธ์ของภาษาคาเฟโอบีเจ
- ขั้นตอนการตรวจสอบความถูกต้องของข้อกำหนด ได้ทำการตรวจสอบความถูกต้องโดยการใส่ข้อมูลเข้าแบบโพสต์ฟิกเข้าไปและตรวจสอบผลลัพธ์ซึ่งได้ออกมาอยู่ในรูปของอินฟิกที่ถูกต้องดังรูปที่ 5.18

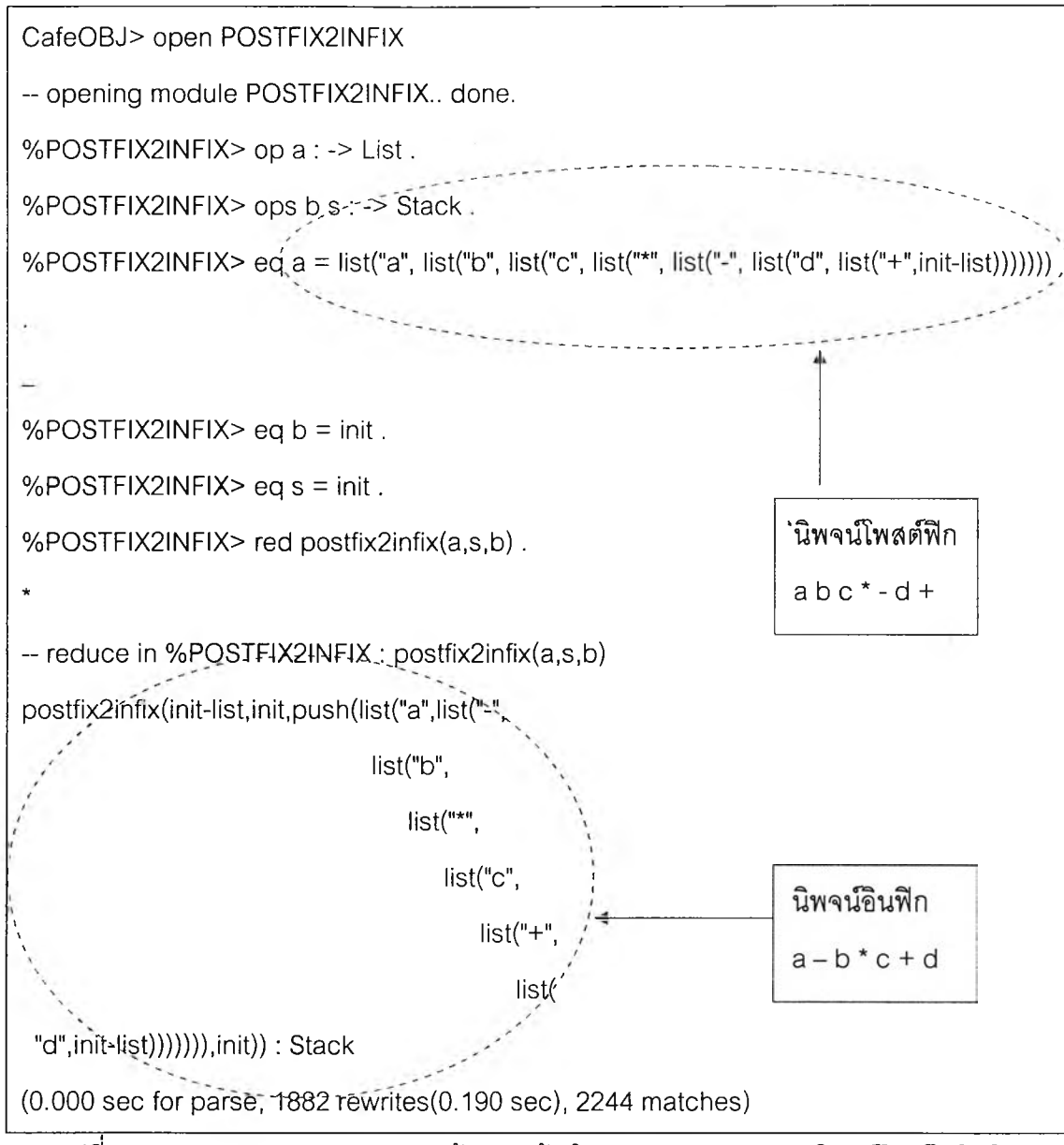
```
mod* POSTFIX2INFIX {
pr( ISMEMBER + LIST + STACK-OF-LIST )
op postfix2infix : List Stack Stack -> Stack

var P  : List
var I : Stack
var S : Stack

ceq postfix2infix(P, S, I) = postfix2infix(tail(P), S, push(list(head(P),init-list), I)) if
isoperand(head(P)) == true and empty(P) == false .
ceq postfix2infix(P, S, I) = postfix2infix(P, push(top(I),S), pop(I)) if isoperator(head(P))
== true and isempty(S) == true and empty(P) == false .
ceq postfix2infix(P, S, I) = postfix2infix(tail(P), pop(S), push(merge(top(I),
merge(list(head(P), init-list), top(S))), pop(I))) if isoperator(head(P)) == true and
isempty(S) == false and empty(P) == false .
}
```

รูปที่ 5.16 ข้อกำหนดคาเฟโอบีเจของการแปลงนิพจน์โพสต์ฟิกเป็นอินฟิก

```
CafeOBJ> in postfix2infix
processing input : .\postfix2infix.mod
-- defining module* POSTFIX2INFIX_*...._...*
done.
CafeOBJ>
```

รูปที่ 5.17 ผลการตรวจสอบวากยสัมพันธ์ของมอดูลการแปลงโพสต์ฟิกเป็นอินฟิก

จากรูปที่ 5.18 ได้ดำเนินการตรวจสอบความถูกต้องของข้อกำหนด ได้กำหนดให้กรณีทดสอบของนิพจน์โพสต์ฟิกอยู่ในรูปของ abc*+d- และผลลัพธ์ที่ได้จากการแปลงซึ่งอยู่ในรูปของนิพจน์อินฟิกคือสมการ c*b+a-d



รูปที่ 5.18 ผลการตรวจสอบความถูกต้องของข้อกำหนดมอดูลการแปลงโพสต์ฟิกเป็นอินฟิก

5.2.3 กรณีศึกษาที่ 3 การเรียงลำดับแบบแทรก (Insertion sort)

การเรียงลำดับข้อมูล คือขบวนการจัดเตรียมกลุ่มของสารสนเทศให้อยู่ในลักษณะที่เรียงลำดับจากน้อยไปมาก (Ascending order) หรือจากมากไปน้อย (Descending order)

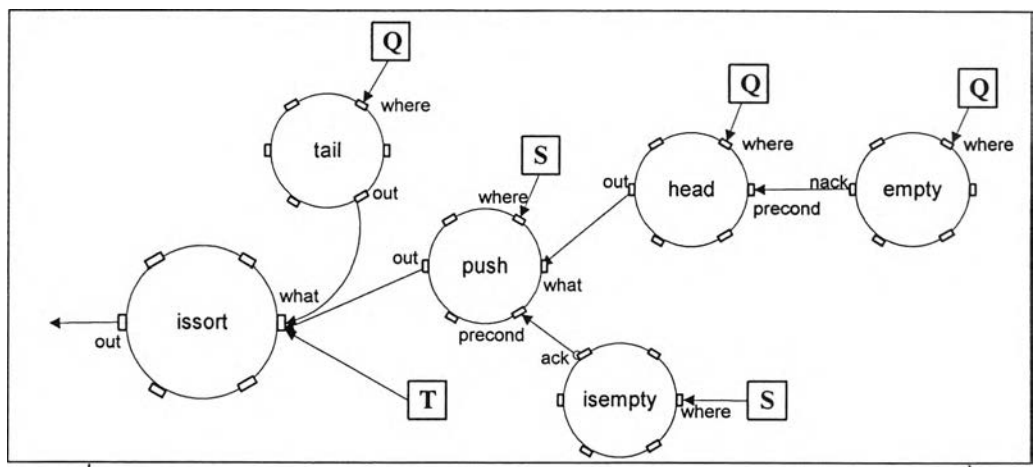
การเรียงลำดับแบบแทรก [14] เป็นการแทรกระเบียนข้อมูลในตำแหน่งที่เรียงลำดับกันภายในกลุ่มของข้อมูลชุดนั้น เพื่อจะวางตำแหน่งของข้อมูลให้ถูกต้องตามเงื่อนไขของการเรียงลำดับ ดำเนินการซ้ำจนกระทั่งข้อมูลของสมาชิกที่ยังไม่ได้พิจารณานั้นถูกแทรกข้อมูลเพิ่มเข้ามาในกลุ่มที่เรียงลำดับแล้ว จนข้อมูลในกลุ่มมีครบเท่ากับข้อมูลตั้งต้น

วิธีการเรียงลำดับแบบแทรก จะใช้ข้อมูลเริ่มต้นอยู่ในรูปของโครงสร้างแบบแถวคอย (แต่ในการวิจัยนี้จะใช้โครงสร้างข้อมูลแบบรายการแทน เนื่องจากมีโครงสร้างข้อมูลที่เหมือน

กัน) ผลลัพธ์จะเก็บอยู่ในรูปแบบของโครงสร้างข้อมูลแบบกองซ้อน และจะมีกองซ้อนอีกแถวหนึ่งเป็นที่พักข้อมูล โดยในงานวิจัยนี้ได้เลือกทำการเรียงลำดับจากน้อยไปหามาก ซึ่งเมื่อพิจารณาการเรียงแบบนี้ทีละขั้นได้ สามารถสรุปออกมาเป็นเงื่อนไขการเรียงได้ 6 กรณี และเมื่อสร้างเป็นแผนภาพเครือข่ายอนุภาคความต้องการได้ดังนี้

กรณีที่ 1 ถ้ากองซ้อนที่เก็บผลลัพธ์ยังไม่มีข้อมูล ให้อ่านข้อมูลเริ่มต้นตัวแรกไปเก็บไว้ยังกองซ้อนสำหรับผลลัพธ์

จากรูปที่ 5.19 ซึ่งเป็นแผนภาพเครือข่ายอนุภาคความต้องการที่แสดงการเรียงลำดับแบบแทรกกรณีที่ 1 เมื่อทำการตรวจสอบข้อมูลของโครงสร้างแบบรายการ Q (โหนด empty) ว่ามีข้อมูลอยู่จริงแล้ว จะทำการตรวจสอบโครงสร้างข้อมูลแบบกองซ้อน S (โหนด isempty) ถ้าเป็นกองซ้อนที่ว่าง จะนำส่วนหัวของโครงสร้างข้อมูลแบบรายการ Q มาใส่ (โหนด push) เข้าไปยังโครงสร้างข้อมูลแบบกองซ้อน S ดังนั้นข้อมูลที่อยู่ที่โครงสร้างข้อมูลแบบรายการ Q จะเหลือเพียงส่วนหาง (โหนด tail) เท่านั้น

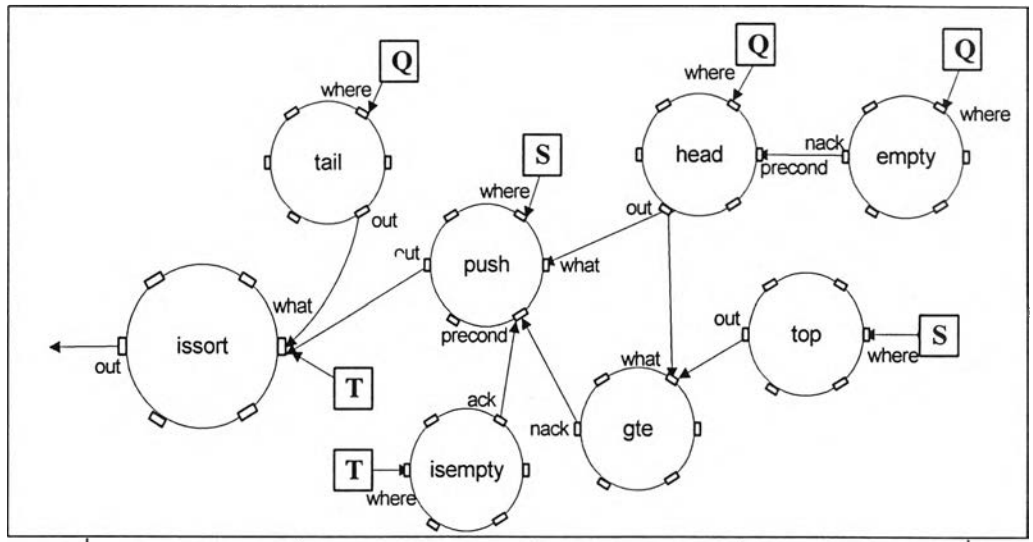


รูปที่ 5.19 แผนภาพเครือข่ายอนุภาคความต้องการแสดงการเรียงแบบแทรกกรณีที่ 1

กรณีที่ 2 ถ้ากองซ้อนสำหรับพักข้อมูลว่าง และข้อมูลเข้าที่อ่านเข้ามามีค่าน้อยกว่าส่วนบนสุดของกองซ้อนที่เก็บผลลัพธ์ ให้ทำการเก็บค่าที่อ่านเข้ามาไปยังกองซ้อนที่เก็บผลลัพธ์

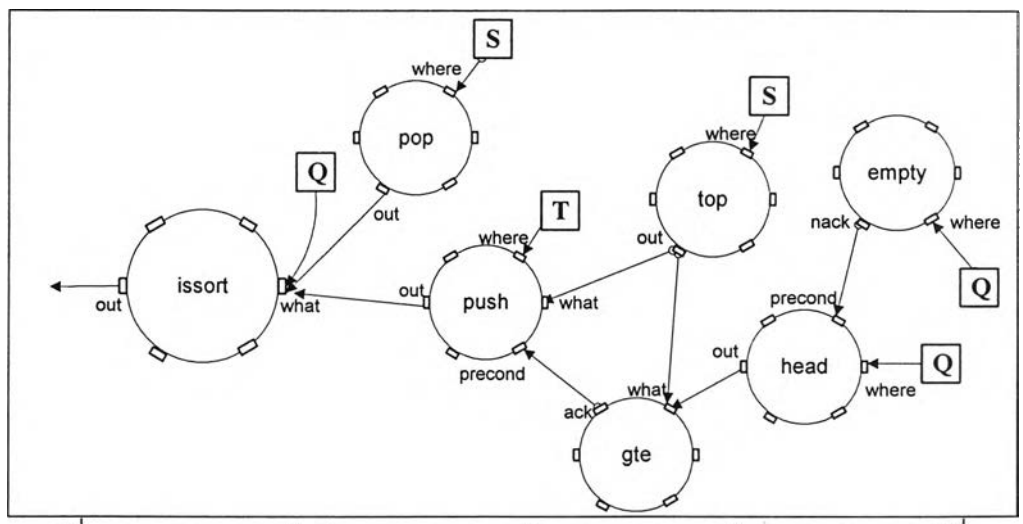
จากรูปที่ 5.20 ซึ่งเป็นแผนภาพเครือข่ายอนุภาคความต้องการที่แสดงการเรียงลำดับแบบแทรกกรณีที่ 2 เมื่อทำการตรวจสอบข้อมูลของโครงสร้างแบบรายการ Q (โหนด empty) ว่ามีข้อมูลอยู่แล้ว และทำการตรวจสอบโครงสร้างข้อมูลแบบกองซ้อน T (โหนด isempty) ถ้าเป็นกองซ้อนที่ว่าง จะนำส่วนหัวของโครงสร้างข้อมูลแบบรายการ Q (โหนด head) มาทำการเปรียบเทียบค่าความสำคัญกับส่วนบนสุดของโครงสร้างข้อมูลแบบกองซ้อน S (โหนด gte) และถ้าผล

ออกมาเป็นเท็จ จะนำส่วนหัวของโครงสร้างข้อมูลแบบรายการ Q มาใส่ (โหนด push) เข้าไปยังโครงสร้างข้อมูลแบบกองซ้อน S ดังนั้นข้อมูลที่อยู่ที่โครงสร้างข้อมูลแบบรายการ Q จะเหลือเพียงส่วนหาง (โหนด tail) เท่านั้น



รูปที่ 5.20 แผนภาพเครือข่ายอนุภาคความต้องการแสดงการเรียงแบบแทรกกรณีที่ 2

กรณีที่ 3 ถ้าค่าที่อ่านเข้ามามีค่ามากกว่าส่วนบนสุดของกองซ้อนที่เก็บผลลัพธ์ ให้นำค่าที่อยู่บนสุดของกองซ้อนที่เก็บผลลัพธ์ ไปเก็บไว้ยังกองซ้อนสำหรับพักข้อมูล

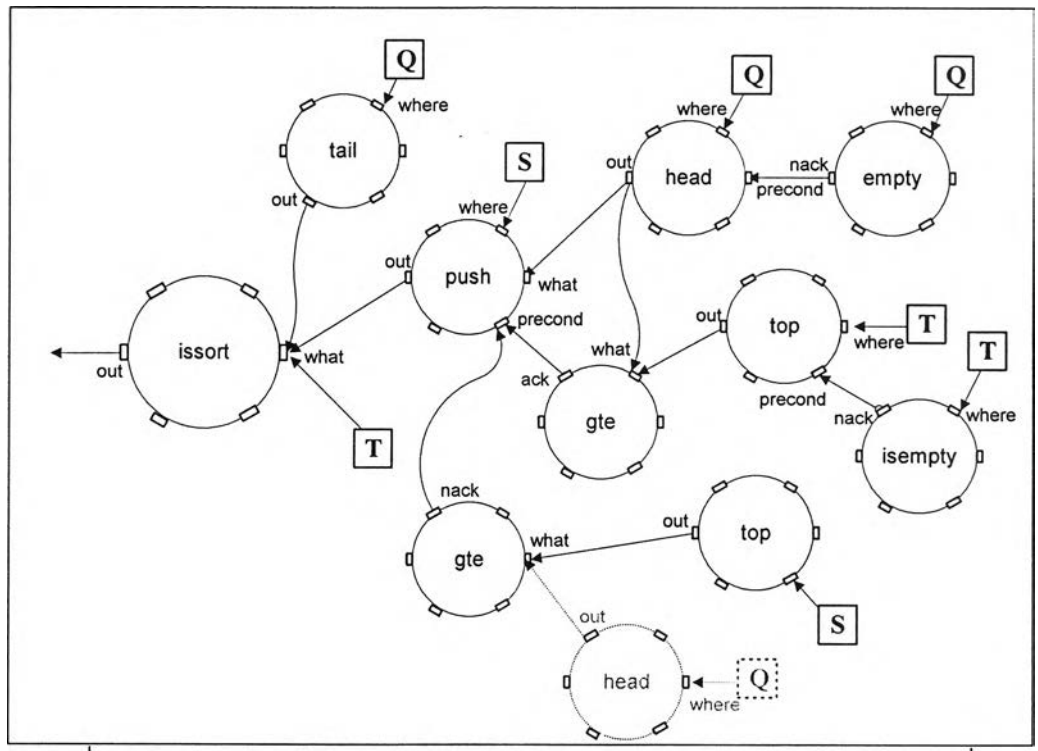


รูปที่ 5.21 แผนภาพเครือข่ายอนุภาคความต้องการแสดงการเรียงแบบแทรกกรณีที่ 3

จากรูปที่ 5.21 ซึ่งเป็นแผนภาพเครือข่ายอนุภาคความต้องการที่แสดงการเรียงลำดับแบบแทรกกรณีที่ 3 เมื่อทำการตรวจสอบข้อมูลของโครงสร้างแบบรายการ Q (โหนด empty) ว่ามีข้อมูลอยู่แล้ว จะนำส่วนหัวของโครงสร้างข้อมูลแบบรายการ Q (โหนด head) มาทำการ

เปรียบเทียบค่าความสำคัญกับส่วนบนสุดของโครงสร้างข้อมูลแบบกองซ้อน S (โหนด gte) และถ้าผลออกมาเป็นจริง จะนำส่วนบนสุดของโครงสร้างข้อมูลแบบกองซ้อน S มาใส่ (โหนด push) เข้าไปยังโครงสร้างข้อมูลแบบกองซ้อน T ดังนั้นข้อมูลที่อยู่ที่โครงสร้างข้อมูลแบบกองซ้อน S จะถูกลดลง (โหนด pop) ไปหนึ่งค่า

กรณีที่ 4 ถ้ากองซ้อนสำหรับพักข้อมูลและผลลัพธ์ไม่ว่าง และค่าที่อ่านเข้ามามีค่าน้อยกว่าค่าบนสุดของกองซ้อนสำหรับผลลัพธ์ แต่มากกว่าค่าบนสุดของกองซ้อนสำหรับพักข้อมูล ให้นำค่าที่อ่านเข้ามาเก็บไว้ที่กองซ้อนสำหรับผลลัพธ์

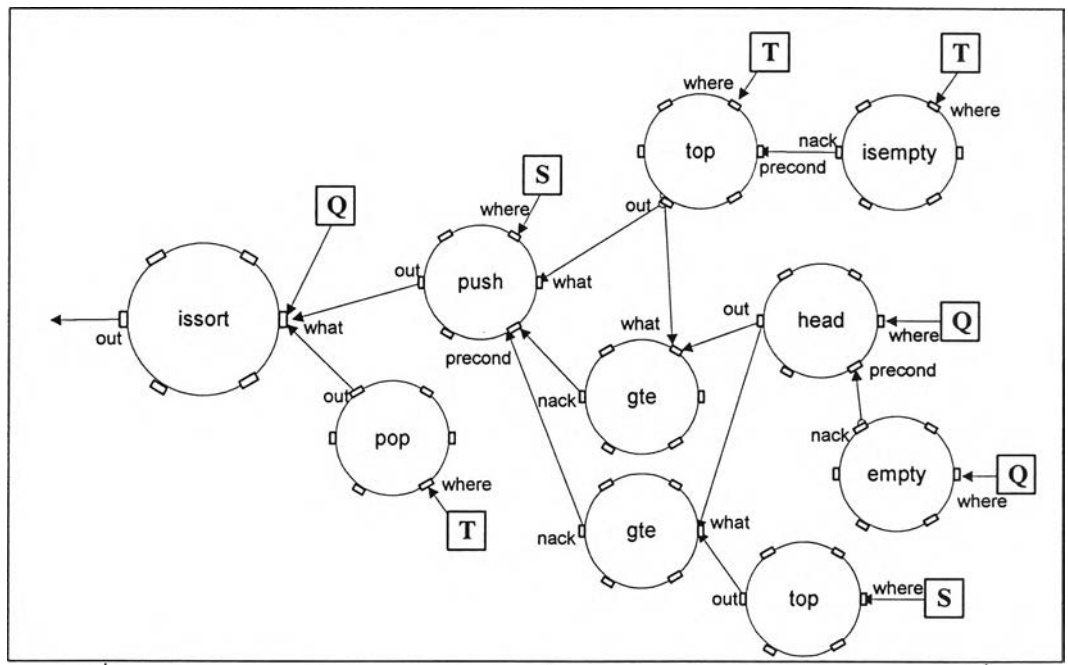


รูปที่ 5.22 แผนภาพเครือข่ายอนุภาคความต้องการแสดงการเรียงแบบแทรกกรณีที่ 4

จากรูปที่ 5.22 ซึ่งเป็นแผนภาพเครือข่ายอนุภาคความต้องการที่แสดงการเรียงลำดับแบบแทรกกรณีที่ 4 เมื่อทำการตรวจสอบข้อมูลของโครงสร้างแบบรายการ Q (โหนด empty) ว่ามีข้อมูลอยู่แล้ว จะทำการตรวจสอบข้อมูลที่อยู่ในโครงสร้างแบบกองซ้อน T (โหนด isempty) ถ้ามีข้อมูลจะนำส่วนบนสุดของโครงสร้างข้อมูลแบบกองซ้อน T (โหนด top) มาทำการเปรียบเทียบค่าความสำคัญ (โหนด gte) กับส่วนหัวของโครงสร้างข้อมูลแบบรายการ Q (โหนด head) และทำการเปรียบเทียบค่าความสำคัญของส่วนหัวโครงสร้างข้อมูลแบบรายการ Q กับส่วนบนสุดของโครงสร้างข้อมูลแบบกองซ้อน S (โหนด gte) ถ้าส่วนหัวของโครงสร้างข้อมูลแบบรายการ Q มากกว่าส่วนบนสุดของโครงสร้างแบบกองซ้อน T แต่น้อยกว่าส่วนบนสุดของโครงสร้างแบบกอง

ซ้อน S จะนำส่วนหัวของโครงสร้างข้อมูลแบบรายการ Q มาใส่ (โหนด push) เข้าไปยังโครงสร้างข้อมูลแบบกองซ้อน T ดังนั้นข้อมูลที่อยู่ที่โครงสร้างข้อมูลแบบรายการ Q จะเหลือเพียงส่วนหาง (โหนด tail) เท่านั้น

กรณีที่ 5 ถ้ากองซ้อนสำหรับพักข้อมูลและผลลัพธ์ไม่ว่าง และค่าที่อ่านเข้ามามีค่าน้อยกว่าค่าบนสุดของทั้งกองซ้อนสำหรับผลลัพธ์ และกองซ้อนสำหรับพักข้อมูลแล้ว ให้อ่านค่าบนสุดของกองซ้อนสำหรับพักข้อมูลมาเก็บไว้ยังกองซ้อนสำหรับผลลัพธ์



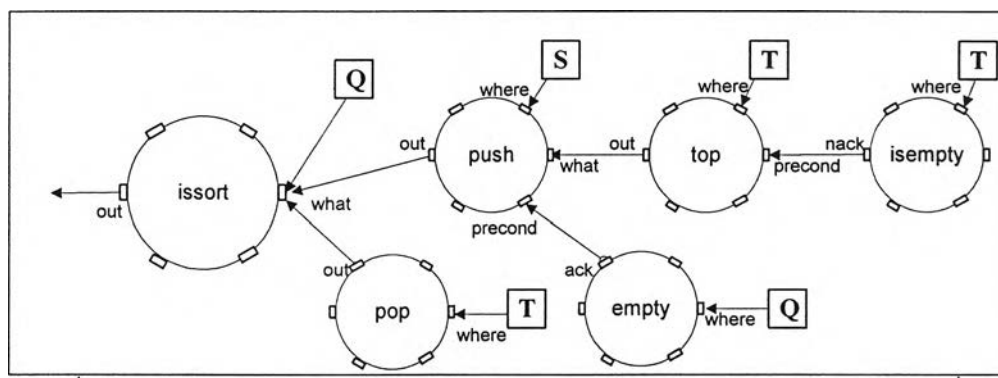
รูปที่ 5.23 แผนภาพเครือข่ายอนุภาคความต้องการแสดงการเรียงแบบแทรกกรณีที่ 5

จากรูปที่ 5.23 ซึ่งเป็นแผนภาพเครือข่ายอนุภาคความต้องการที่แสดงการเรียงลำดับแบบแทรกกรณีที่ 5 เมื่อทำการตรวจสอบข้อมูลของโครงสร้างแบบรายการ Q (โหนด empty) ว่ามีข้อมูลอยู่แล้ว จะทำการตรวจสอบข้อมูลที่อยู่ในโครงสร้างแบบกองซ้อน T (โหนด isempty) ถ้ามีข้อมูลจะนำส่วนบนสุดของโครงสร้างข้อมูลแบบกองซ้อน T (โหนด top) มาทำการเปรียบเทียบค่าความสำคัญ (โหนด gte) กับส่วนหัวของโครงสร้างข้อมูลแบบรายการ Q (โหนด head) และทำการเปรียบเทียบค่าความสำคัญของส่วนหัวโครงสร้างข้อมูลแบบรายการ Q กับส่วนบนสุดของโครงสร้างข้อมูลแบบกองซ้อน S (โหนด gte) ถ้าส่วนหัวของโครงสร้างข้อมูลแบบรายการ Q มีค่าน้อยกว่าส่วนบนสุดของโครงสร้างแบบกองซ้อน T และส่วนบนสุดของโครง สร้างแบบกองซ้อน S แล้วจะนำส่วนบนสุดของโครงสร้างข้อมูลแบบกองซ้อน T มาใส่ (โหนด push) เข้าไปยังโครง

สร้างข้อมูลแบบกองซ้อน S ดังนั้นข้อมูลที่อยู่ที่โครงสร้างข้อมูลแบบกองซ้อน T จะถูกลดลง (โหนด pop) ไปหนึ่งค่า

กรณีที่ 6 ถ้ามีข้อมูลเริ่มต้นหมดแล้ว แต่กองซ้อนสำหรับพักข้อมูลยังมีข้อมูลค้างอยู่ ให้ทำการอ่านค่าที่อยู่บนสุดของกองซ้อน มาไว้ในกองซ้อนสำหรับผลลัพธ์จนหมด

จากรูปที่ 5.24 ซึ่งเป็นแผนภาพเครือข่ายอนุภาคความต้องการที่แสดงการเรียงลำดับแบบแทรกกรณีที่ 6 เมื่อทำการตรวจสอบข้อมูลของโครงสร้างแบบรายการ Q (โหนด empty) ว่าไม่มีข้อมูลเหลืออยู่แล้ว จะทำการตรวจสอบโครงสร้างข้อมูลแบบกองซ้อน T (โหนด isempty) ถ้ามีข้อมูลอยู่จะนำส่วนบนสุดของโครงสร้างข้อมูลแบบกองซ้อน T (โหนด top) มาใส่ (โหนด push) เข้าไปยังโครงสร้างข้อมูลแบบกองซ้อน S ดังนั้นข้อมูลที่อยู่ที่โครงสร้างข้อมูลแบบกองซ้อน T จะถูกลดลง (โหนด pop) ไปหนึ่งค่า



รูปที่ 5.24 แผนภาพเครือข่ายอนุภาคความต้องการแสดงการเรียงแบบแทรกกรณีที่ 6

จากรูปเครือข่ายอนุภาคความต้องการที่ได้ เมื่อนำมาแปลงให้อยู่ในรูปของอาร์พีเอ็นเท็กซ์เพื่อจะทำเป็นข้อมูลนำเข้า จะได้อาร์พีเอ็นเท็กซ์ดังภาคผนวก ข และเมื่อทำการแปลงเป็นข้อกำหนดคาเฟโอบีเจ โดยใช้คลังจัดเก็บตัวดำเนินการจากภาคผนวก ค จะได้ผลลัพธ์ดังรูปที่ 5.25

การตรวจสอบความถูกต้องของมอดูลการทำการเรียงลำดับแบบแทรก ได้ทำการตรวจสอบโดยนำข้อกำหนดที่ได้ใส่ไปในโปรแกรมการแปลคาเฟโอบีเจ ซึ่งผลการตรวจสอบทั้งสองส่วนมีดังนี้

- ขั้นตอนการตรวจสอบความถูกต้องตามวากยสัมพันธ์ของภาษาคาเฟโอบีเจ จะได้ผลดังรูปที่ 5.26 ซึ่งแสดงว่าข้อกำหนดของมอดูลการเรียงลำดับแบบแทรกที่สร้างจากแผนภาพเครือข่ายอนุภาคความต้องการถูกต้องตามวากยสัมพันธ์ของภาษาคาเฟโอบีเจ

```
mod* ISSORT {
pr( LIST + STACK + PRECEDENCE )

op issort : List Stack Stack -> Stack

var Q : List
var S : Stack
var T : Stack

ceq issort(Q, S, T) = issort(tail(Q), push(head(Q), S), T) if isempty(S) == true and
empty(Q) == false .
ceq issort(Q, S, T) = issort(tail(Q), push(head(Q), S), T) if gte(head(Q), top(S)) == false
and empty(Q) == false and isempty(T) == true .
ceq issort(Q, S, T) = issort(Q, pop(S), push(top(S), T)) if gte(head(Q), top(S)) == true
and empty(Q) == false .
ceq issort(Q, S, T) = issort(tail(Q), push(head(Q), S), T) if isempty(T) == false and
gte(head(Q), top(T)) == true and gte(head(Q), top(S)) == false and empty(Q) == false .
ceq issort(Q, S, T) = issort(Q, push(top(T), S), pop(T)) if isempty(T) == false and
empty(Q) == false and gte(head(Q), top(T)) == false and gte(head(Q), top(S)) == false .
ceq issort(Q, S, T) = issort(Q, push(top(T), S), pop(T)) if empty(Q) == true and
isempty(T) == false .
}
```

รูปที่ 5.25 ข้อกำหนดคาเฟโอบีเจของการทำการเรียงลำดับแบบแทรก

- ขั้นตอนการตรวจสอบความถูกต้องของข้อกำหนด ได้ทำการตรวจสอบความถูกต้องโดยการใส่ข้อมูลที่ยังไม่ได้ทำการเรียงลำดับเข้าไปและตรวจสอบผลลัพธ์ซึ่งได้ออกมาอยู่ในรูปของข้อมูลที่เรียงลำดับจากน้อยไปมากที่ถูกต้องดังรูปที่ 5.27

```
CafeOBJ> in insertsort
processing input : .\insertsort.mod
-- defining module* ISSORT_*....._.....* done.
CafeOBJ>
```

รูปที่ 5.26 ผลการตรวจสอบวากยสัมพันธ์ของมอดูลการเรียงลำดับแบบแทรก

จากรูปที่ 5.27 ได้ดำเนินการตรวจสอบความถูกต้องของข้อกำหนด ได้กำหนดให้กรณีทดสอบเป็นตัวอักษรที่เรียงกันดังนี้ "f c d b a j i" และหลังจากทำการเรียงลำดับใหม่แล้วได้ผลลัพธ์ออกมาดังนี้ "a b c d f i j"

```
CafeOBJ> open ISSORT
-- opening module ISSORT.. done.
%ISSORT> op q  : -> List .
%ISSORT> ops s t : -> Stack .
%ISSORT> eq q = list("f", list("c", list("d", list("b", list("a",list("j",list("i",init-list)))))))) .
_
%ISSORT> eq s = init .
%ISSORT> eq t = init .
%ISSORT> red issort(q,s,t) .
*
-- reduce in %ISSORT : issort(q,s,t)
issort(init-list,push("a",push("b",push("c",push("d",
                                push("f",
                                    push("i",
                                        push("j",
                                            init))))))),
    init) : Stack
(0.000 sec for parse, 276639 rewrites(8.763 sec), 308650 matches)
```

กรณีทดสอบ "f c d b a j i"

ผลการเรียงลำดับ "a b c d f i j"

รูปที่ 5.27 ผลการตรวจสอบความถูกต้องของข้อกำหนดมอดูลการเรียงลำดับแบบแทรก

5.3 ระบบที่ใช้ทดสอบโปรแกรม

ระบบที่ใช้ในการทดสอบโปรแกรม ได้ทดสอบภายใต้ระบบปฏิบัติการไมโครซอฟท์วินโดวส์ 2000 สำหรับเครื่องคอมพิวเตอร์ที่ใช้ในการทดสอบโปรแกรม มีคุณสมบัติดังนี้

- เครื่องคอมพิวเตอร์แบบพกพา CPU Pentium III 733 MHz.
- หน่วยความจำหลัก 128 MB
- ความจุฮาร์ดดิสก์ 20 GB
- ระบบปฏิบัติการไมโครซอฟท์วินโดวส์ 2000