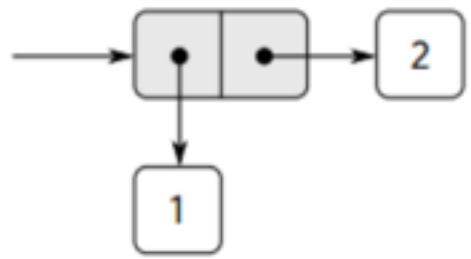
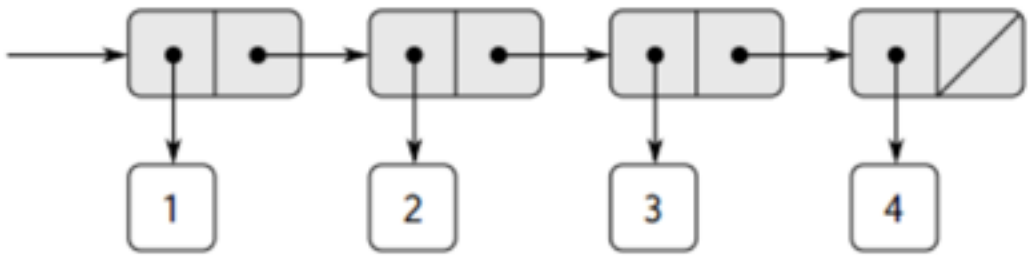
3.3 변형 가능한 데이터로 프로그래밍 하기
Modeling with Mutable Data

@cgryu

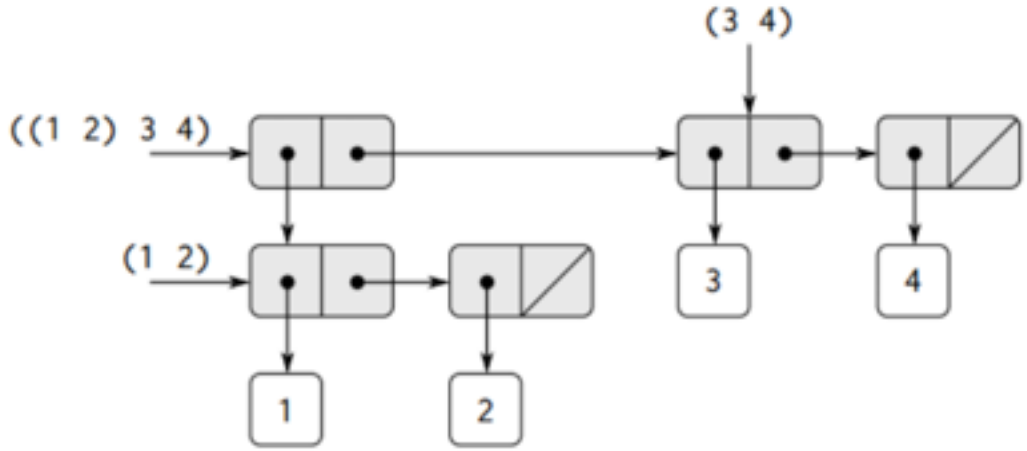
- 3.3.1 변형 가능한 리스트
- 3.3.2 큐
- 3.3.3 표
- 3.3.4 디지털 회로 시뮬레이터
- 3.3.5 관계 알리기(constraint propagation)



(cons 1 2)



(cons 1 (cons 2 (cons 3 (cons 4 '()))))
(list 1 2 3 4)

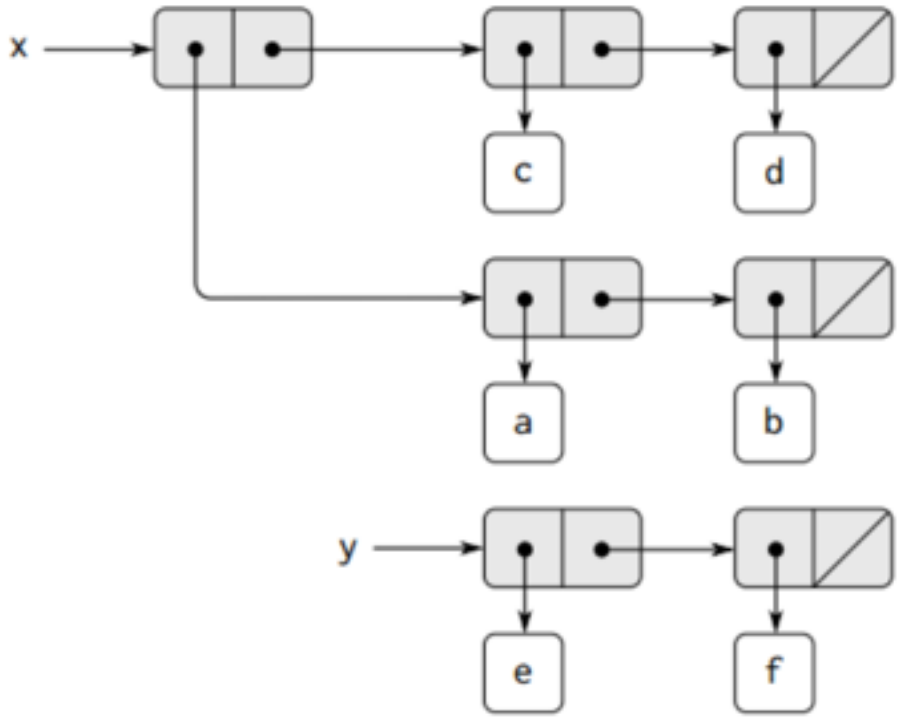


(cons (list 1 2) (list 3 4))

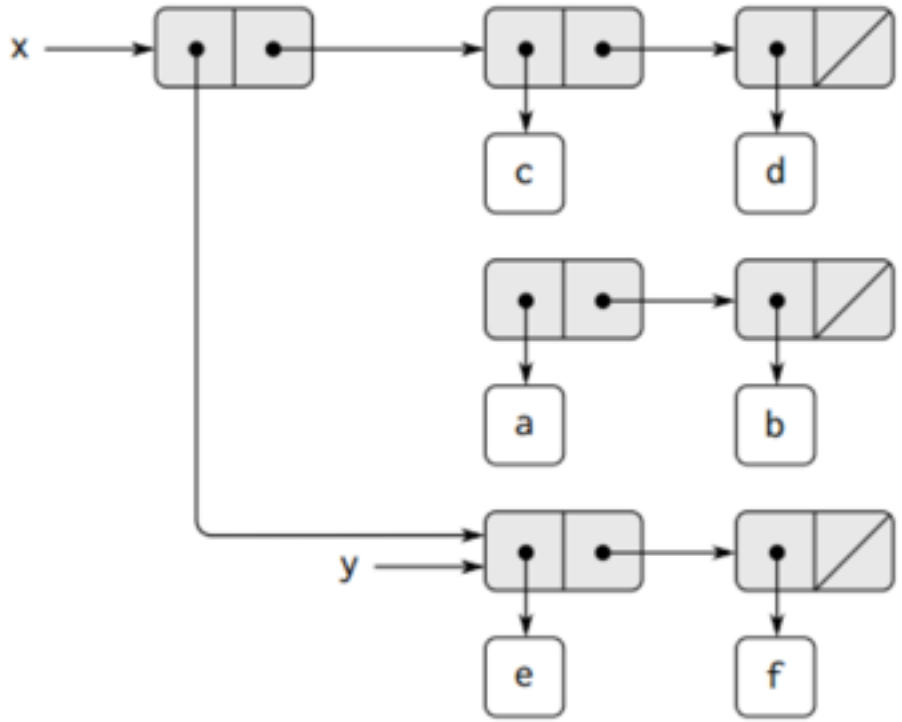
- **3.3.1 변형 가능한 리스트**

(set-car! <pair> <obj>)

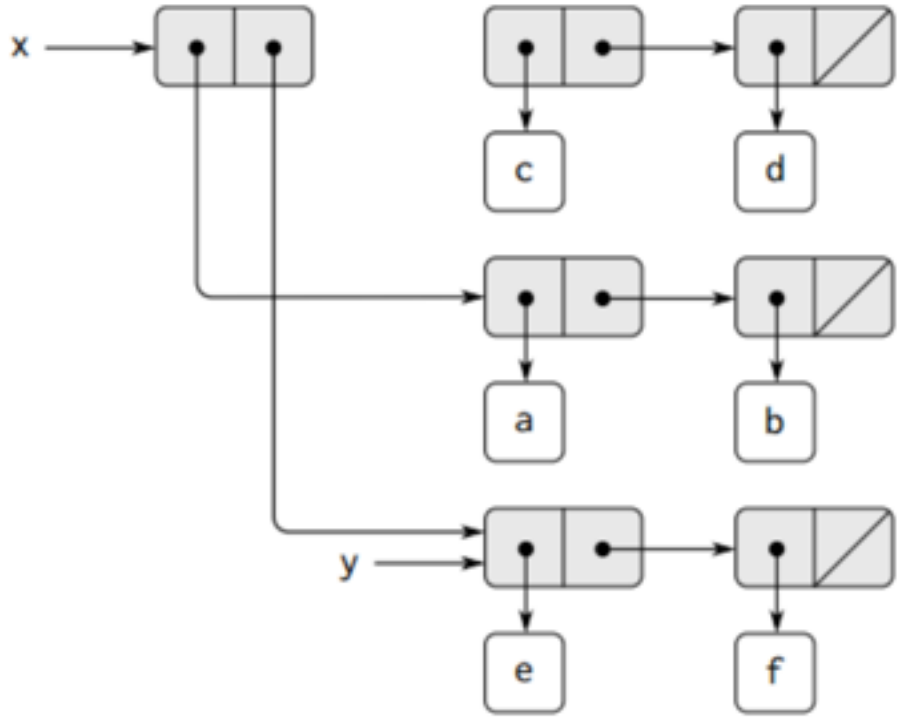
(set-cdr! <pair> <obj>)



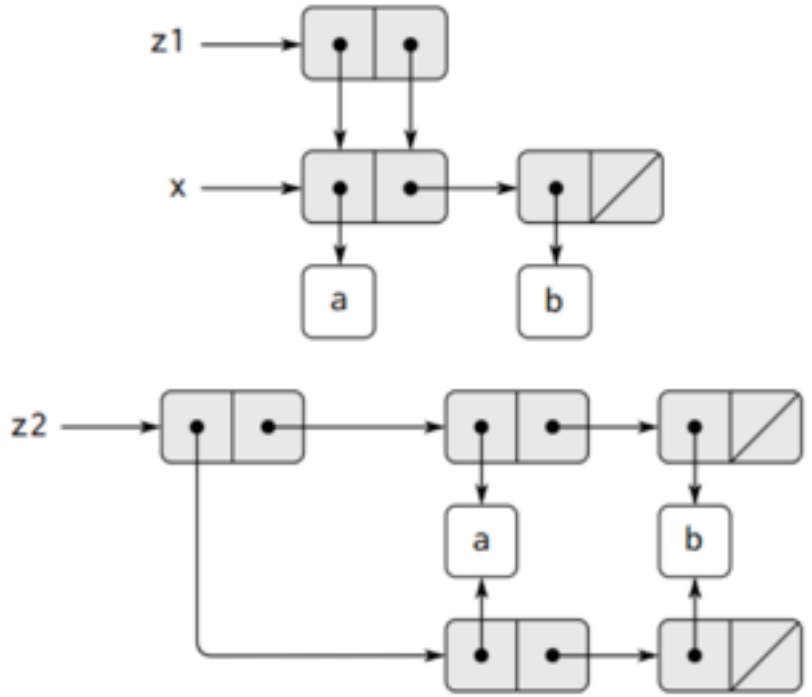
(define x (list (list 'a 'b) 'c 'd))
(define y (list 'e 'f))



(set-car! x y)



(set-cdr! x y)




```
(define x (list 'a 'b))
(define z1 (cons x x))
(define z2 (cons (list 'a 'b) (list 'a 'b)))
```

```
(define (set-to-wow! x) (set-car! (car x) 'wow) x)

(set-to wow! z1)
((wow b) wow b)

(set-to wow! z2)
((wow b) a b)
```

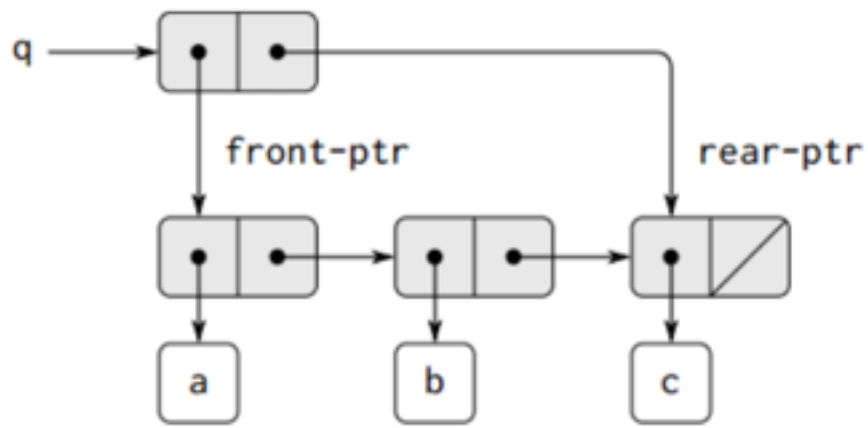
```
(define x (list 'a 'b))
(define z1 (cons x x))
(define z2 (cons (list 'a 'b) (list 'a 'b)))

(eq? z1 z2)
(eq? (car z1) (cdr z1))
(eq? (car z2) (cdr z2))

#f
#t
#f
```

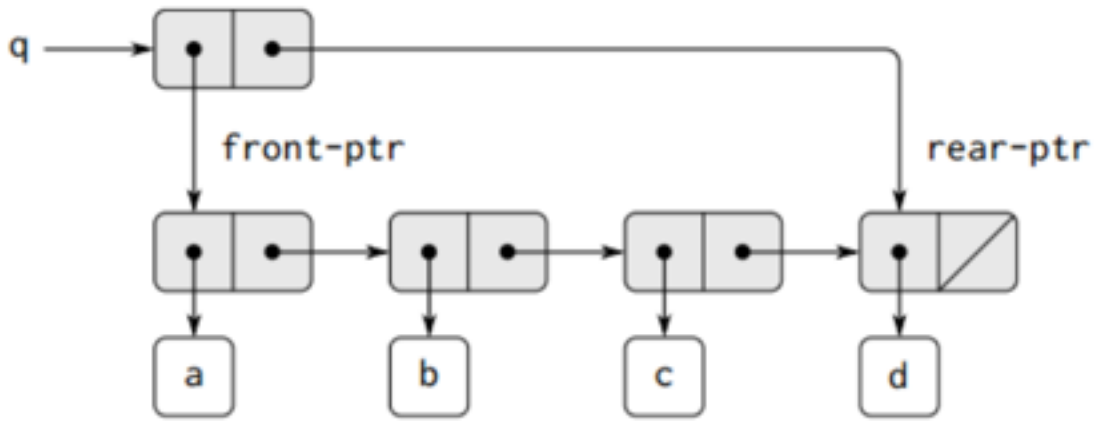
- **3.3.2 큐**

큐란, 뒤(rear)로 원소를 집어넣고 앞(front)으로 꺼낼 수 있는 차례열이다

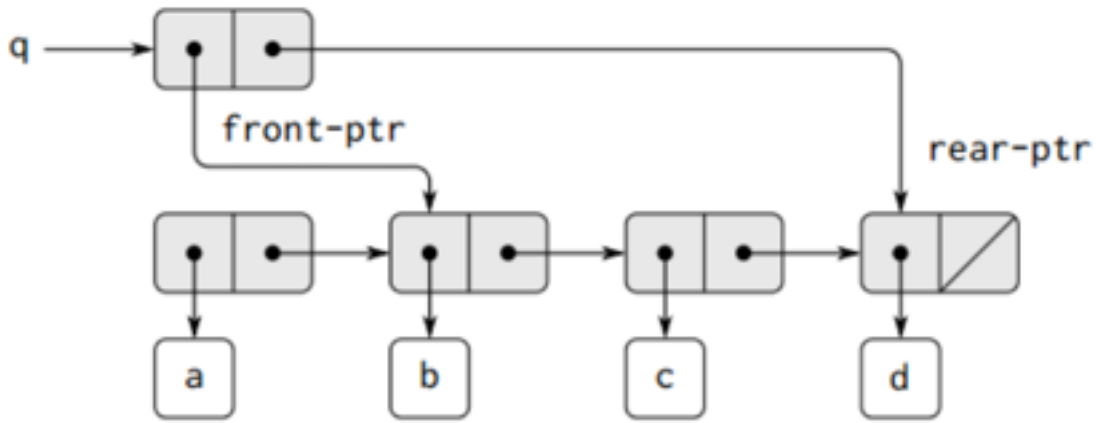
FIFO(First In First Out)

(make-queue)
(empty-queue? <queue>)
(front-queue <queue>)
(insert-queue! <queue> <item>)
(delete-queue! <queue>)


q
front-ptr
rear-ptr
a
b
c



(insert-queue! q ‘d)



(delete-queue! q ‘d)

- **3.3.3 표**

키로 데이터를 찾아보기 쉽게 관리

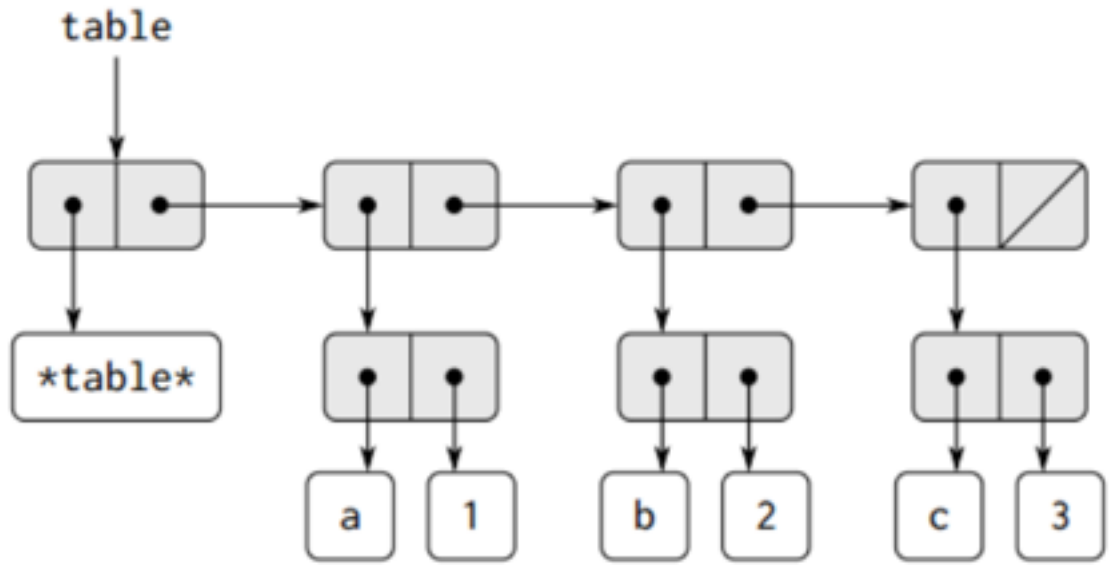


table
table
a
1
b
2
c
3

```
(define (lookup key table)
  (let ((record (assoc key (cdr table))))
    (if record
        (cdr record)
        false)))
(define (assoc key records)
  (cond ((null? records) false)
        ((equal? key (caar records)) (car records))
        (else (assoc key (cdr records)))))
(define (insert! key value table)
  (let ((record (assoc key (cdr table))))
    (if record
        (set-cdr! record value)
        (set-cdr! table
                  (cons (cons key value)
                        (cdr table)))))
    'ok)
(define (make-table)
  (list '*table*))
```

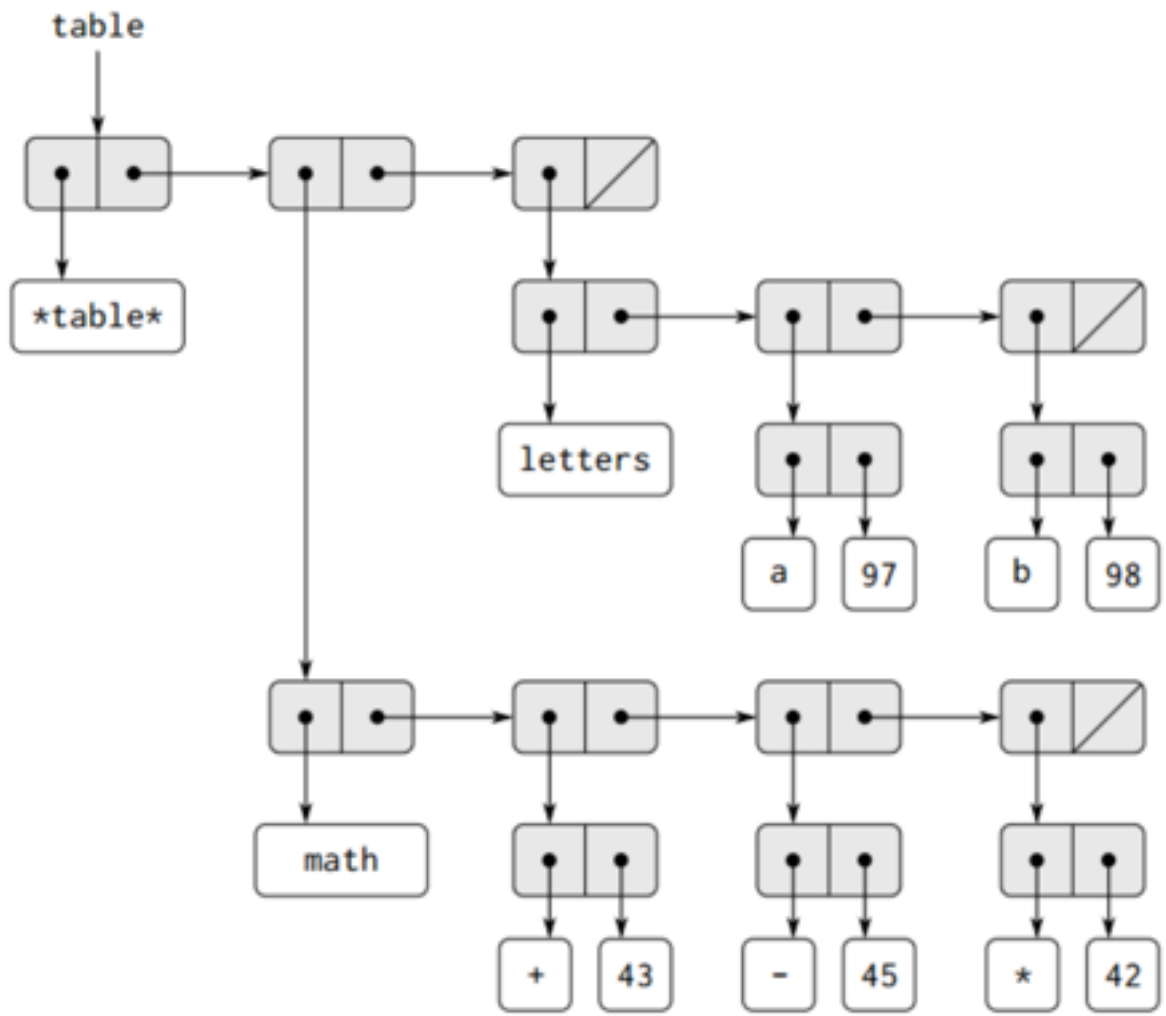
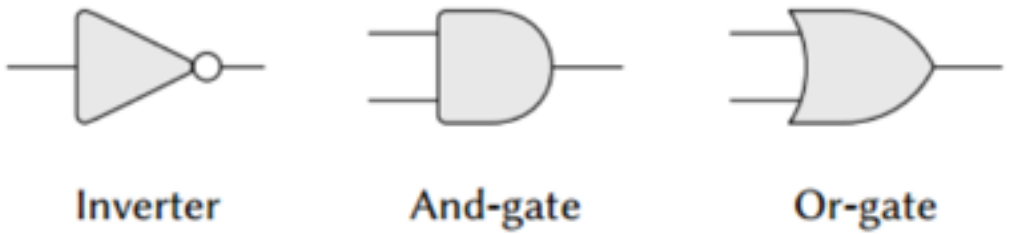


table
table
letters
a
97
b
98
math
+
43
-
45
*
42

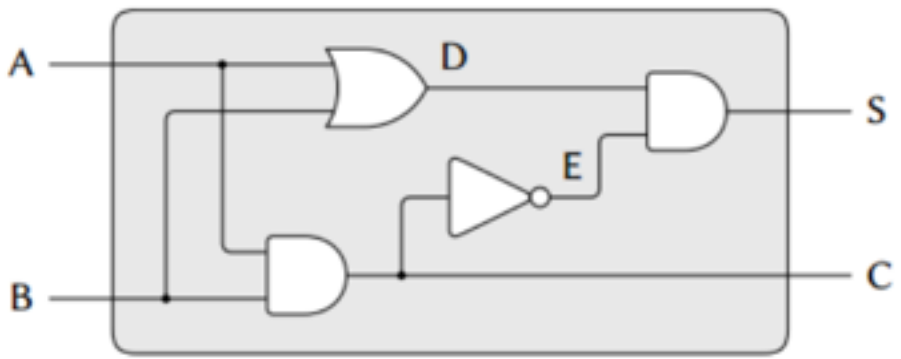
- **3.3.4 디지털 회로 시뮬레이터**

사건 중심 시뮬레이션 (event-driven simulation)

사건중심이란, 여러 가지 움직임이 뒤에 일어날 사건을 일으키는 원인이 되고 다시 그 사건이 더 많은 사건을 일으키며 맞물려 돌아가도록 프로그램을 짜는 일

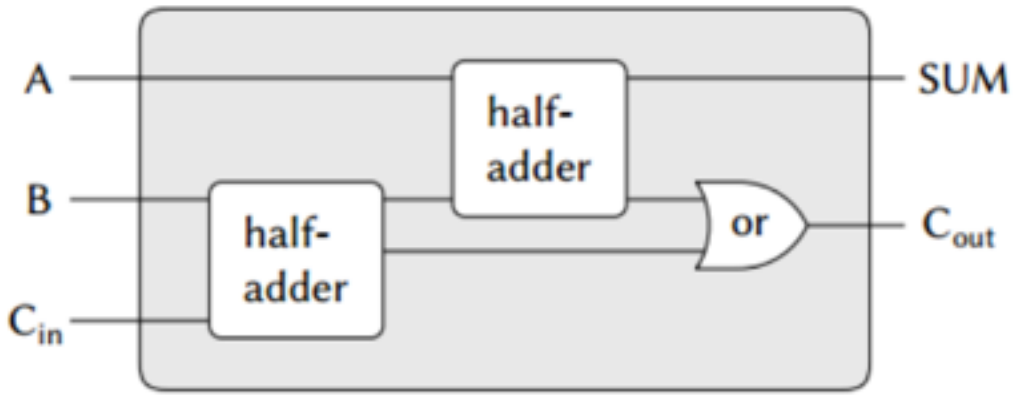


인버터(뒤집개), 논리곱, 논리합

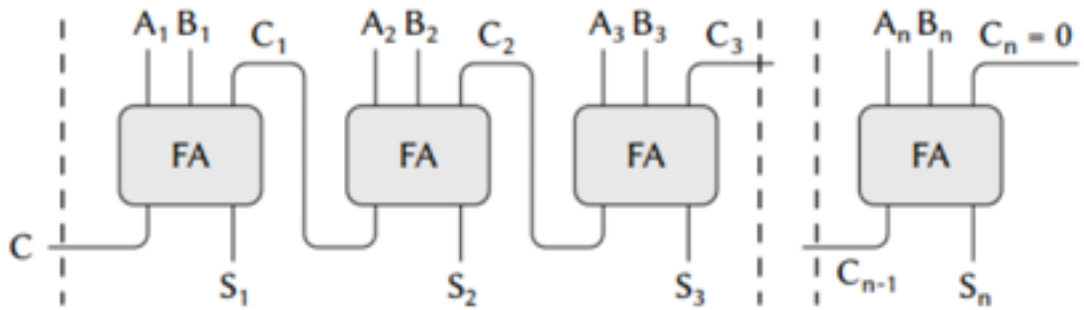


반 덧셈기 회로 (half-adder)

```
(define a (make-wire))
(define b (make-wire))
(define c (make-wire))
(define d (make-wire))
(define e (make-wire))
(define s (make-wire))
(or-gate a b d)
ok
(and-gate a b c)
ok
(inverter c e)
ok
(and-gate d e s)
ok
(define (half-adder a b s c)
  (let ((d (make-wire)) (e (make-wire)))
    (or-gate a b d)
    (and-gate a b c)
    (inverter c e)
    (and-gate d e s)
    'ok))
```



온 덧셈기 회로 (full-adder)



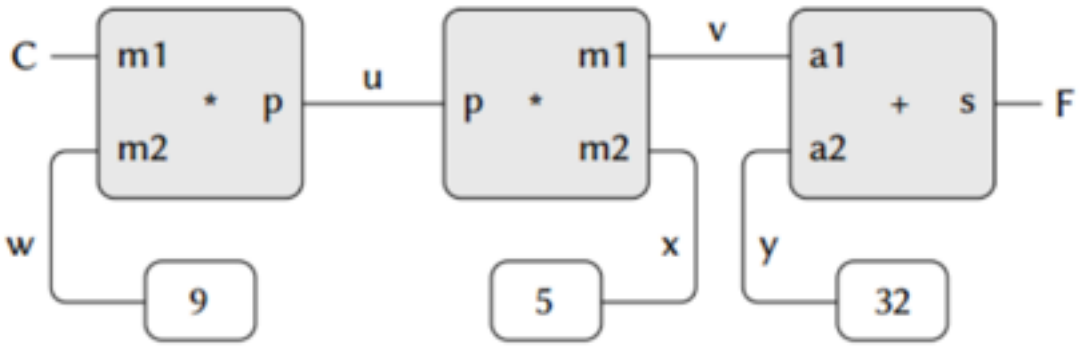
N비트 수를 더하는 자리 올림 덧셈기

- **3.3.5 관계 알리기(constraint propagation)**

물체에 미치는 힘의 관계

쇠막대가 휘는 정도 d
막대에 미치는 힘 f
막대 길이 L
자름 넓이(단면적) A
탄성 계수(탄성 곁수) E

$$dAE = FL$$



화씨온도와 섭씨온도
9C = 5(F - 32)

```
(set-value! C 25 'user)
Probe: Celsius temp = 25
Probe: Fahrenheit temp = 77
done

(set-value! F 212 'user)
Probe: Fahrenheit temp = 212
Probe: Celsius temp = 100
done
```

```
(define C (make-connector))
(define F (make-connector))

(celsius-fahrenheit-converter C F)
ok

(define (celsius-fahrenheit-converter c f)
  (let ((u (make-connector))
        (v (make-connector))
        (w (make-connector))
        (x (make-connector))
        (y (make-connector)))
    (multiplier c w u)
    (multiplier v x u)
    (adder v y f)
    (constant 9 w)
    (constant 5 x)
    (constant 32 y)
    'ok))
```

끝