CSP

Communicating Sequential Processes

Altprogram

- CSP dilinde rekürsif olmayan bir altprogram paralel çalışan bir proses olarak gerçeklenir.
- Ancak, altprogram sadece bir tek prosese hizmet verebilir.
- Kendisini çağıran prosesten ayrıktır.
- Kendisini çağıran proses ile paralel yürütülür.
- Parametre değerlerini giriş işlemi (?) ile kabul eder.
- Sonuçları çıkış işlemi (!) ile gönderir.

Altprogram

[subr::SUBROUTINE || X:: USER]

SUBROUTINE :

*[X?(value param) → …; X!(result param)]

USER:

subr!(arguments); …; subr?results;

Çokgirişli Altprogram

Çokgirişli (multiple entry): yinelemeli bir deyim içinde, her girişi temsil eden ayrı bir seçenek bulunur. Her seçenek bir giriş işlemi ile korunur. Giriş işlemi yapısal özelliğe sahip bir veri okur; kurucu adı seçenekleri birbirinden ayırır.

SUBROUTINE :

* [X?entry1(value param) → …
  [] X?entry2(value param) → …
  ]

USER:

subr! entry1(arguments1); …; subr?results;

veya subr! entry2(arguments2); …; subr?results;

Örnekler

- Kalanlı bölme işlemi gören fonksiyon.

```
[  DIV::*[x,y:integer; X?(x,y) →
                quot, rem: integer; quot := 0; rem := x;
                *[rem >= y → rem := rem – y; quot := quot +1];
                  X!(quot,rem)
            ]
        || X::USER
]
```

- X prosesi sona erince, DIV prosesi de sona erer.

Rekürsif Altprogram

- Rekürsif altprogram bir proses dizisi ile simüle edilir.
- Her proses bir rekürsif çağrı düzeyine karşı düşer; ilk çağrıyı yapan USER prosesi düzey 0 olacaktır.
- Her proses kendinden önce gelen prosesten parametrelerini alır ve sonuçlarını ona iletir. Gerek duyarsa, kendini izleyen prosesi çağırır.

```
[recsub(0)::USER||recsub(i:1..reclimit)::RECSUB]

USER:recsub(1)!(arguments); …;recsub(1)!(results);
```

- Dilin statik yapısı nedeniyle, rekürsif çağrı düzeyine sabit bir üst sınır gereklidir (proses dizisinin boyutu).

Örnek: Rekürsif Faktoriyel

- Rekürsif yöntemle faktoriyel hesabı.

```
[fact(i:1..limit)::
*[n:integer; fact(i-1)?n →
    [ n=0 → fact(i-1)!1
    [] n > 0 → fact(i+1)! n-1;
       r:integer; fact(i+1)?r; fact(i-1)!(n*r)
    ]]
|| fact(0)::USER
]
```

fact(0) sona erince, fact(1) koruması başarısız olduğu için sona erecektir. fact(2), benzer şekilde, fact(1) sonlandığı için sona erecektir. Zincirleme hareket ile tüm prosesler sonlanır.

Örnek: tamsayı kümesi

- Max. 100 elemanlı bir tamsayı kümesini S prosesi temsil etmektedir. S, kümede eleman arama ve ekleme işlemlerini yerine getirir.

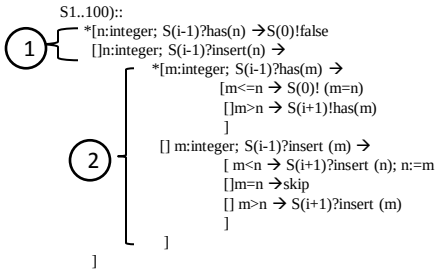
```
S::content:(0..99)integer;size:integer;size:=0;
*[n:integer; X?has(n) →SEARCH;X!(i<size)
  []n:integer; X?insert(n) →SEARCH;
          [i<size → skip
          []i=size; size<100 →
                content(size):=n;size:=size+1
          ]]
SEARCH:i:integer; i:=0;* [i<size ; content(i)<>n →i:=i+1]
```

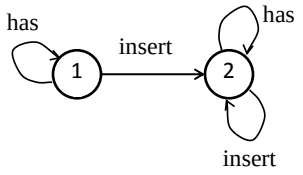
Örnek: rekürsif veri gösterilimi

- Max. 100 elemanlı bir tamsayı kümesi S proses dizisi ile temsil edilmektedir. Her proses bir küme elemanına sıralı şekilde sahiptir (P_n kümenin n. sıradaki elemanı).

```
S(1..100)::
*[n:integer; S(i-1)?has(n) →S(0)!false
 []n:integer; S(i-1)?insert(n) →
                *[m:integer; S(i-1)?has(m) →
                    [m<=n → S(0)! (m=n)
                     []m>n → S(i+1)!has(m)
                    ]
                 [] m:integer; S(i-1)?insert (m) →
                    [ m<n → S(i+1)!insert (n); n:=m
                    []m=n →skip
                    [] m>n → S(i+1)!insert (m)
                    ]
                 ]
]
```

Örnek: rekürsif veri gösterilimi





Küme değerlerini üreten S0 prosesi:
S(0): S(1)!has(5)
[[i:1..100) S(i)?b → skip]

- S(0) sona erince tüm prosesler sona erer.
- Ard arda gönderilen insert istekleri paralel yürütülür.

Monitör'ün Proses Olarak Gerçeklenmesi

- Monitör, birden fazla kullanıcı proses ile haberleşen bir proses olarak gerçeklenir.
- Her kullanıcı proses farklı bir ad veya indise sahip olmalıdır ve her kullanıcı ile kurulan iletişim kaynak/hedefi kesin olarak belirlemelidir.
- Monitör proses kullanıcılarından herhangi biri ile haberleşmek için (hangisi önce çağrı üretmiş ise) indis aralıklı bir korumalı deyim kullanır:

  *[(i:1..100) X(i)?(value param) →…; X(i)!results]

- X proses dizisi içinde çıkış deyimi hazır olanlardan herhangi biri seçilir, kabul edilen giriş param uygun işlemler yapılır ve "i" indisinin belirlediği kullanıcı prosesine sonuçlar iletilir.

Monitör'ün Proses Olarak Gerçeklenmesi

- Monitör, belirli koşullar altında bazı kullanıcı proseslerden giriş kabul etmek istemez ise, giriş deyiminden önce uygun bir lojik koruma deyimi eklenebilir:
- Örnek: j:=0;

  *[(i:1..100) i <> j; X(i)?(value param) →
  …; j:=i; X(i)!results]

- Bu örnekte, monitör bir kullanıcı prosesten gelecek iki çağrıyı arda arda kabul etmeyecektir; ikinci çağrı arada bir başka prosesin çağrısı kabul edilip, işlenene kadar bekletilecektir.
- Ayrıca, yine koruma koşulları getirilerek, giriş işlemleri monitör içinde bulunduğu durum çağrının kabulüne uygun olana kadar geciktirilebilir. Böylece koşul değişkenlerine de gerek duyulmaz.

Üretici-Tüketici Problemi

- Tampon alanı yöneten monitör: proses X
- Üretici proses: producer:: X!p
- Tüketici proses: consumer:: X!more(); X?p
- Proses X:

```
X::
buffer:(0..9) portion;
in, out:integer; in:=0; out:=0;
*[ in< out+10; producer?buffer(in mod 10) →in:=in+1
 [] out< in; consumer?more() →consumer!buffer(out mod 10);
                                          out:=out+1
 ]
```

Üretici-Tüketici Problemi

```
*[ in< out+10; producer?buffer(in mod 10) →in:=in+1
  [] out< in; consumer?more() →consumer!buffer(out mod 10);
                                          out:=out+1
 ]
```

- Koruma durumları:

i. out<in<out+10: buffer alanında boş yer de var, çekilebilecek eleman da; X her türlü isteğe yanıt verebilir. Hangisi önce davranırsa, veya ikisi de hazır ise içlerinden rastgele birinin mesajı alınacaktır.

ii. in=out: buffer alanı boş; "out<in" koruması yanlış olacağından sadece üretici istekleri yanıtlanır.

iii. in=out+10: buffer alanı dolu; "in<out+10" koruması yanlış olacağından sadece tüketici istekleri yanıtlanır.

- X ne zaman sonlanır?
  - in=out doğru olup da producer prosesinin sona ermesi halinde.
  - in< out+10 doğru ancak producer sonlanmış: koruma FALSE
  - out< in koruması FALSE

Semafor

- Semafor S X(i:1..100) proses dizisinde yer alan prosesler arasında paylaşılmaktadır.
- S!V() semafor değerini 1 artırır.
- S!P() semafor değerini 1 eksiltir ancak semafor değeri pozitif değilse bu çağrıyı yapan proses bekletilmelidir.

```
S:: val:integer; val:=0;
    *[(i:1..100) X(i)? V() → val:=val+1
      [] (i:1..100); val > 0; X(i)? P() → val:=val-1
     ]
```

- Çözümde çağrıyı yapan prosesin indis değeri kullanılmamıştır.
- S prosesi tüm X prosesleri sona erdikten sonra sonlanır.

Sieve of Erathosthenes
Asal Sayı Üreticisi

- Problem:10000 değerinden küçük olan tüm asal sayıları küçükten büyüğe doğru yazdır.
- SIEVE proses dizisinin her elemanı bir önceki prosesten aldığı ilk sayı asal sayıdır. Bu değeri yerel değişkeninde tutar ve çıkışa aktarır. Daha sonra, bir önceki prosesten büyüyen sırada değerler alır, bunlardan kendi asal sayısının bir katı olanları eler, diğerlerini ise kendini izleyen sırada yer alan prosese aktarır.
- Proses dizisinin ilk elemanı (SIEVE(0)) sayı dizisinin üretir.
- print prosesi çıkış prosesi olarak görev yapar ve kendisine gönderilen asal sayıların dış ortama aktarır.

Asal Sayı Üreticisi

```
[SIEVE(i:1..100)::
    p, mp:integer;
    SIEVE(i-1)?p;  // aldığı ilk değer asal sayıdır
    print!p;            // çıkışa aktar
    mp:=p;
    *[m:integer; SIEVE(i-1)?m →
                *[m > mp →mp:=mp+p];  // katını hesapla
                [m = mp → skip
                [] m < mp → SIEVE(i+1)! m  // katı değil
                ]                   // bir sonraki prosese gönder
     ]
```

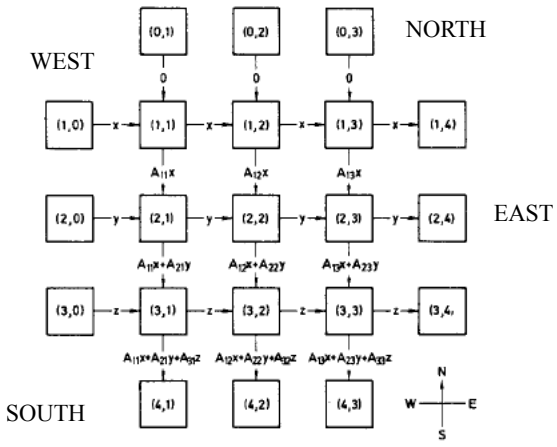
Asal Sayı Üreticisi

```
[
…..
|| SIEVE(0):: print!2; n:integer; n:=3;
                *[ n< 10000 → SIEVE(1)! n; n:=n+2 ]
|| SIEVE(101):: *[ n:integer; SIEVE(100)? n →  print!n]
|| print:: *[(i:0..101) n:integer; SIEVE(i)? n → …(yazdır) ]
]
```

PARALEL MATRİS ÇARPIMI

- A(3,3) matrisi ile IN(3,3) matrisi çarpılacaktır.
- M(i:1..3, j:1..3) proses matrisinde yer alan her proses A matrisinin bir elemanına sahiptir.
  - M(i,j)prosesi: A(i,j) değeri
- IN matrisine ait elemanlar, sütünlar halinde , giriş bilgisi olarak gönderilir. Bu değerler M(i:1..3, 0) prosesleri (WEST) tarafından sağlanır.
- M(i:1..3, j:1..3) prosesleri iki ana işlem yaparlar:
  - Sol komşudan aldıkları IN matrisine ait değeri sağ komşuya aktarmak
  - Üst komşudan aldıkları değere soldan edindikleri değerin A matris elemanıyla çarpımını ekleyip, sonucu alt komşuya göndermek.
- Çarpım sonucu M(4, j:1..3) prosesleri (SOUTH) tarafından okunur.

PARALEL MATRİS ÇARPIMI



PARALEL MATRİS ÇARPIMI

```
[ M(i: 1..3,0)  ::WEST
|| M(0, j: 1..3)    ::NORTH
|| M(i: 1..3,4)     ::EAST
|| M(4, j: 1..3)    ::SOUTH
|| M(i: 1..3, j: 1..3)::CENTER
]
```

- CENTER =

```
*[x:real; M(i,j-1)?x→
          M(i,j+1)!x;
          sum:real;
          M(i-1,j)?sum;
          M(i+1,j)!(A(i,j)*x+sum)
 ]
```

- NORTH = *[true→ M(1,j)!0]
- EAST = *[x:real; M(i,3)?x→skip]