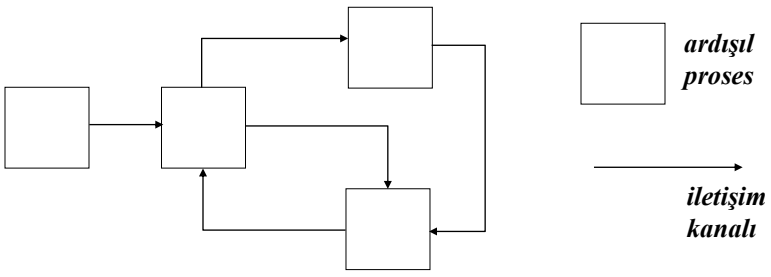
CSP

Communicating Sequential Processes

CSP

- Hoare, C.A.R. Communicating Sequential Processes. *Communications of the ACM 21, 8*. (August 1978), 666-677.



CSP

- Temel özellikler:
  - Paralel Program: Mesaj aktarımına dayalı proses ağı (prosesler bir ortak belleği paylaşmazlar)
  - Input/Output (Giriş/Çıkış) deyimleri-haberleşme ve senkronizasyon sağlamak için kullanılır.
  - Dijkstra'nın korumalı deyimler (guarded commands: G-> CL) kullanılır.
  - Paralel deyimler (cobegin-coend benzeri) mevcuttur.

CSP DİLİNİN TANIMI

CSP sentaks tanımında kullanılan özel işaretler ve anlamları

- ::= tanımı verir
- | veya (bir başka tanım)
- { } içerdiği tanımın sıfır veya daha fazla kez tekrarını öngörür
- ? Input deyimi
- ! Output deyimi
- * çevrim ifade eder
- […] begin-end bloğu ifade eder
- || paralel çalışmayı ifade eder

CSP DİLİNİN TANIMI

- <command> ::= <simple command>|
  <structured command>
- <simple command> ::=
  <null command>|
  <assignment command>|
  <input command>|
  <output command>
- <null command> ::= skip

Atama Deyimi

- <assignment command> ::= <target variable> := <expression>
- <expression> ::= <simple expression>|<structured expression>
- <structured expression> ::= <constructor>(<expression list>)
- <constructor> ::= <identifier>|<empty>
- <expression list> ::= <empty>| <expression>{,<expression>}
- <target variable> ::= <simple variable>|<structured target>
- <structured target> ::= <constructor>(<target variable list>)
- <target variable list> ::= <empty>| <target variable> {,<target variable>}

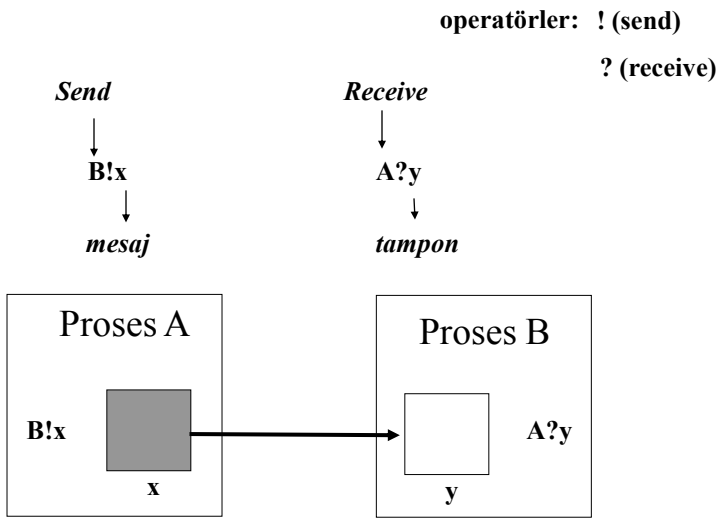
Örnek Atama Deyimleri

(1) x := x+ 1	(1) x değerini 1 arttır.
(2) (x, y) := (y, x)	(2) x ve y değerlerini değiş tokuş eder.
(3) x := cons(left, right)	(3) Yapısal özelliğe sahip bir değer oluşturur ve bu değeri x'e atar.
(4) cons(left, right) := x	(4) Eger "x" *cons(y,* z) yapısında değil ise hata üretir; aksi halde, left:=y ve right:=z atanır.
(5) insert(n) := insert(2*x + 1)	(5) n := 2*x + 1 atamasına eşdeğer.
(6) c := P()	(6) c'e P kurucusuna sahip olan ve bileşenleri bulunmayan bir sinyal atar.
(7) P() := c	(7) c'nin değeri P() değil ise hatalı olur; aksi durumda etkisiz işlemdir.
(8) insert(n) := has(n)	(8) Tanım uyumsuzluğu nedeniyle hata üretir.

Giriş/Çıkış Deyimleri

- <input command> ::= <source>?<target variable>
- <output command> ::= <destination>!<expression>
- <source> ::= <process name>
- <destination> ::= <process name>
- <process name> ::=
  <identifier>|<identifier>(<subscripts>)
- <subscripts> ::= <integer expression>{,<integer expression>}

Giriş/Çıkış Deyimleri



9

Giriş/Çıkış Deyimleri

- G/Ç deyimleri paralel çalışan iki proses arasındaki haberleşmeyi belirtirler.
- Haberleşmenin gerçekleşmesi için:
  - Bir prosesteki input deyimi kaynak (source) olarak bir diğer prosesin adını verir.
  - Diğer proses de output deyiminde hedef olarak kendini kaynak olarak gösteren prosesi tanımlar.
  - Input deyimindeki "target variable" ile output deyiminde yer alan ifade tip açısından uyumludur.

Giriş/Çıkış Deyimleri

- Bu üç koşul gerçekleşirse, sözkonusu G/Ç deyimlerinin birbirleriyle uyumlu olduğu söylenir ve aynı anda yürütülürler. Sonuçta,

  <target variable>:=<expression>

  şeklinde bir atama gerçekleşir.
- Bir input deyimi, kaynak proses sonlanmış ise, bir output deyimi hedef proses sonlanmış ise başarısız olur.

Giriş/Çıkış Deyimlerine Örnekler

(1) cardreader?cardimage
(2) lineprinter!lineimage
(3)P?(x, y)
(4) DIV!(3*a + b, 13)

(5)console(i)?c
(6) console(j-1)!"A"
(7) x(i)?V( )
(8) sem!P( )

(1) cardreader prosesinden bir kayıt oku, bu bilgileri cardimage değişkenine ata
(2) lineprinter prosesine lineimage değişkeninin içeriğini gönder.
(3) P isimli prosesten bir değer çifti oku ve x ve y değişkenlerine ata.
(4) DIV prosesine beliritilen değerlerin gönder.
Not:Eğer DIV prosesi deyim (3) ve P prosesi de deyim (4) bir arada yürütürlerse, sonuç şu atamalara eşdeğer olur: *(x,y)* ~ (3*a + b, 13) (x:=3*a+b;y:= 13).
(5) console proses dizisinin i. elemanından bir bilgi oku ve c değişkenine ata.
(6) console proses dizisinin (j - l). elemanına , "A“ karakterini gönder.
(7) x proses dizisinin i. elemanından V( ) sinyalini bekle; farklı bir sinyal kabul edilmez.
(8) sem prosesine P( ) sinyalini gönder.

Yapısal Deyimler (Structured Commands)

- <structured command> ::=
  <alternative command>|
  <repetitive command>|
  <parallel command>
- <command list> ::= {<declaration>; | <command>;} <command>

Paralel Deyim

- <parallel command> ::= [<process> {||<process>}]
- <process> ::= <process label> <command list>
- <process label> ::= <empty>|<identifier> :: |<identifier>(<label subscript>{,<label subscript>}) ::
- <label subscript> ::= <integer constant>|<range>
- <integer constant> ::= <numeral>|<bound variable>
- <bound variable> ::= <identifier>
- <range> ::= <bound variable>:<lower bound>..<upper bound>
- <lower bound> ::= <integer constant>
- <upper bound> ::= <integer constant>

Paralel Deyim

- <parallel command> ::= [<process> {II<process>}]
- Paralel deyimi oluşturan her proses birbirinden ayrık olmalıdır- ortak değişkenler, paylaşılan bellek yoktur.
- Paralel deyim, içerdiği tüm proseslerin paralel çalışmasını öngörür. Tüm prosesler aynı anda çalışmaya başlarlar ve tümü sonlandıktan sonra, paralel deyim de başarıyla sonlanır.

Paralel Deyime Örnekler

- [cardreader?cardimage || lineprinter!lineimage]
  - Paralel çalışan iki prosesten oluşur.
  - Her iki proses de sonlanınca, deyim sonlanır.
- [west::DISASSEMBLE || x::SQUASH || east::ASSEMBLE]
  - Deyimi oluşturan üç proses (west, x, east) adlarına sahiptirler. “DISASSEMBLE”,”SQUASH” ve “ASSEMBLE” deyim listelerini temsil etmektedirler.

Paralel Deyime Örnekler

- X(i:1..n):: CL →
  X(1)::CL1||X(2)::CL2||… || X(n)::CLn
- [room::ROOM || fork(i:0..4)::FORK || phil(i:0..4)::PHIL]
  - 11 adet paralel proses tanımlar.
  - "room" adlı prosesin çalışması : ROOM deyim listesi
  - fork(0), fork(1),…fork(4) proseslerin çalışması : FORK
    (i indis değeri her bir fork prosesini diğerinden ayırır)
  - Aynı açıklamalar phil prosesleri için de geçerlidir.

Seçenekli Deyim (Alternative Command)

- <alternative command> ::=
  [<guarded command>(□<guarded command>}]
- <guarded command>::=<guard> →,<commandlist> | (<range>{,<range>})<guard> → <command list>
- <guard> ::= <guard list> ; <input command>
- <guard list>::=<guard element>(;<guard element>}
- <guard element>::=<boolexpression>|<declaration>

Seçenekli Deyime Örnek

- [x >= y → m := x
  □ y >= x → m := y]

- Koruması doğru olan seçeneklerden herhangi biri rastgele seçilir ve işaret ettiği deyimler yürütülür.

- eğer x >= y, m'e x ata;
  eğer y >= x, m'e y ata;
  eğer x >= y ve y >= x ifadelerinin her ikisi de doğru ise (x=y), seçeneklerden herhangi biri seçilir ve yürütülür.

Yinelemeli Deyim (Repetitive Command)

- <repetitive command> ::=
  *<alternative command>

- Yinelemeli deyim kendisini oluşturan seçenekli deyimi mümkün olan en yüksek sayıda yürütülmesini öngörür.
  - Eğer tüm korumalar başarısız olurlarsa, seçenekli deyim başarısız olur→ yinelemeli deyim sona erer.
  - Aksi durumda, korumalı deyimlerden herhangi biri seçilir ve yürütülür; tüm deyim tekrarlanır.

Örnekler

- Problem: *size* boyutunda olan *content* dizisinde "n" değerini ara.
- Çözüm:

  SEARCH:: i:=0; *[i<size;content(i)<>n -> i:=i+1]
- Koruma: i<size VE content(i) <>n
- Korumanın başarısız olması için

  i>=size VEYA content(i) = n olmalı
- Deyim, i=0,1,2… değerleri için (i<size olduğu sürece) ve content(i) <>n olduğu sürece yinelenecektir.

Örnekler

*[(i:1..10)continue (i); console(i)?c → X!(i,c);
console(i)!ack(),
continue(i):=(c<>sign_off);]

- 10 adet koruma: continue (i) değeri TRUE olan ve çıkış işlemi hazır olan herhangi bir console prosesinden c oku. "i" değişkeni hangi console prosesinden veri okunduğunu belirleyecektir. X prosesine (i,c) değer çiftini gönder, console prosesine ack işareti gönder, ve okunan "c" değeri sign-off bilgisine eşit ise, continue(i) FALSE ata (bu prosesten artık giriş yapılmaz). Dizinin tüm elemanları FALSE olunca deyim sona erer.

Örnekler

```
*[  n:integer; X?insert (n) → INSERT
 □ n:integer; X?has(n) → SEARCH; X! (i<size)
 ]
    SEARCH:: i:=0; *[i<size;content(i)<>n -> i:=i+1]
```

- İki seçenekli yinelemeli deyim. Seçeneklerin hangisinin seçileceği X prosesinin gönderdiği mesaja bağlı.
- X prosesi sona erince, deyim de sonlanır.

Örnekler

Problem: semafor görevi yapan proses:

```
sem:: val:integer; val:=0;
*[ X? V ( ) → val:=val+1
 □ val>0 ; Y ? P( ) → val:=val-1
 ]
```

```
Proses X: sem ! V( );
Proses Y: sem ! P( );
```

- Sem prosesi, val>0 olduğu sürece korumalı deyimlerden herhangi biri yürütülebilir. İkinci seçenek sadece val>0 ise seçilebilir.
- Yinelemeli deyiminin sonlanma koşulu:
  - X ve Y proseslerinin her ikisi de sonlanır, veya
  - val<=0 iken X prosesi sonlanır

Örnekler

- Problem: “west” prosesinin ürettiği char verisini “east” prosesine aktaran X prosesini yazın.
- Çözüm:

  X :: *[c:character; west ? c → east ! c]
- *west* prosesi sona erince, “west?c” giriş işlemi başarısız olur ve yinelemeli deyimin sonlanmasına neden olur. Deyim sonlanınca da X prosesi sonlanır.
- X prosesi east ve west prosesleri arasında tek elemanlı bir tampon görevi görmektedir. Böylece west, east’in son üretilen veriyi okumasını beklemeden yeni veri üretimi için çalışmaya başlayabilir.

Örnekler

- Bir önceki problemi, west prosesinden okuyacağı birbirini izleyen çift yıldız (“**”) yerine, east prosesine “↑” gönderecek şekilde değiştirin.

```
X :: *[c:character; west ? c →
        [c <> "*" → east ! c
        [] c= "*" → west ? c ;
                [c <> "*" → east ! "*";   east ! c
                [] c= "*" → east ! "↑"
                ]
        ]
      ]
```