PUCRS – Escola Politécnica – Fernando Luís Dotti - Copyright – direitos reservados. Para uso exclusivo do aluno. Proibida cópia, reprodução e compartilhamenbtio.

Fundamentos de Processamento Paralelo e Distribuído ou
Sistemas Operacionais

Introdução à Concorrência

processos, estados, transições, interleaving, justiça

Referência: Principles of Concurrent and Distributed Programming (Second Edition) Addison-Wesley, 2006. Mordechai (Moti) Ben-Ari

Fernando Luís Dotti

PUCRS | ESCOLA POLITÉCNICA

PUCRS – Escola Politécnica – Fernando Luís Dotti - Copyright – direitos reservados. Para uso exclusivo do aluno. Proibida cópia, reprodução e compartilhamenbtio.

Concorrência – Definições Básicas

Concorrência – Definições Básicas

Um programa concorrente consiste de um conjunto finito de processos *sequenciais*.

Um processo contém um conjunto finito de comandos *atômicos*.

Cada processo tem seu ***control pointer*** que indica o próximo comando que pode ser executado pelo processo.

PUCRS – Escola Politécnica – Fernando Luís Dotti - Copyright – direitos reservados. Para uso exclusivo do aluno. Proibida cópia, reprodução e compartilhamenbtio.

Definições

Um programa concorrente consiste de um conjunto finito de processos *sequenciais*.

Um processo contém um conjunto finito de comandos *atômicos*.

Cada processo tem seu ***control pointer*** que indica o próximo comando que pode ser executado pelo processo.

PUCRS – Escola Politécnica – Fernando Luís Dotti - Copyright – direitos reservados. Para uso exclusivo do aluno. Proibida cópia, reprodução e compartilhamenbtio.

Definições

Uma **computação** descreve uma execução possível do programa concorrente.

Uma computação é obtida por um **entrelaçamento (*interleaving*)** arbitrário dos comandos atômicos dos processos do programa.

O conjunto de todas computações do programa é chamado de **comportamento**.

PUCRS – Escola Politécnica – Fernando Luís Dotti - Copyright – direitos reservados. Para uso exclusivo do aluno. Proibida cópia, reprodução e compartilhamenbtio.

Definições

Semântica de *interleaving* é uma forma de representar a execução de um sistema concorrente.

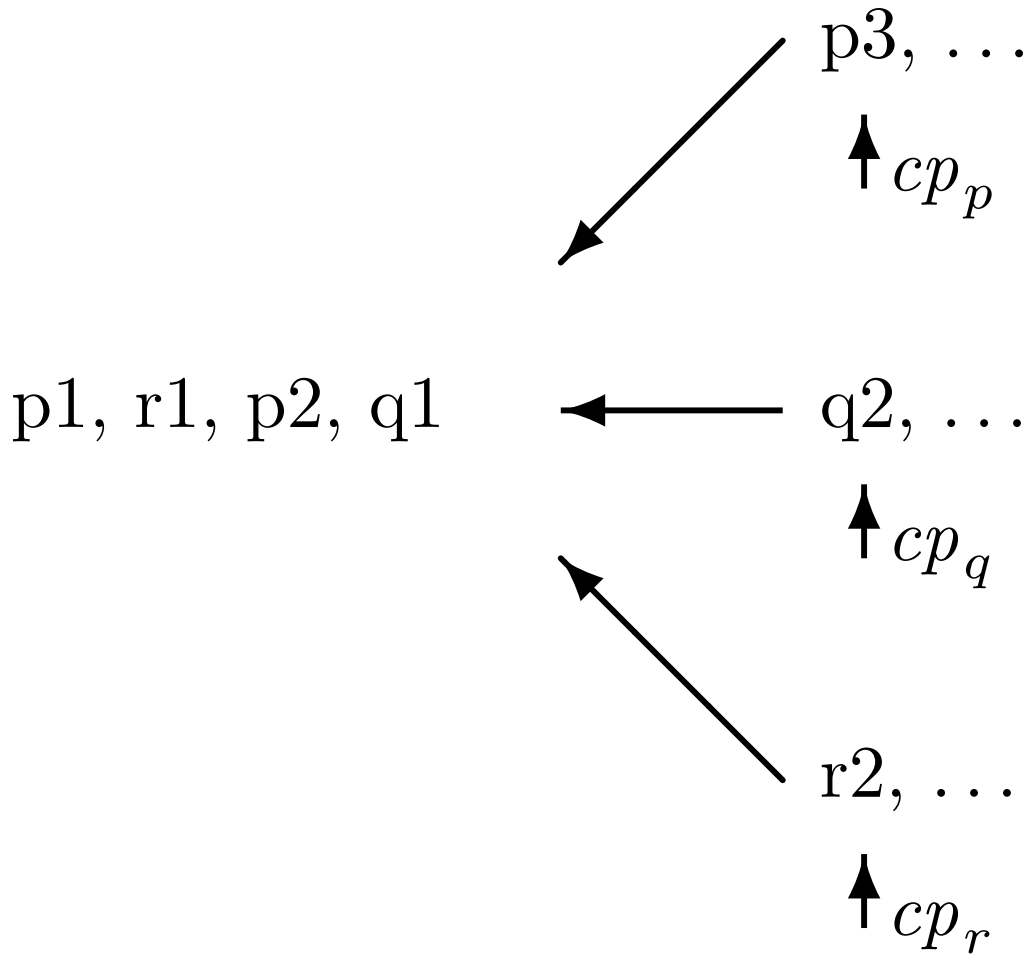
É a mais utilizada na literatura, e adotada aqui.

Há outras semânticas para representar concorrência, como: true concurrency; semântica de eventos; traces de Marzukievicz.

PUCRS – Escola Politécnica – Fernando Luís Dotti - Copyright – direitos reservados. Para uso exclusivo do aluno. Proibida cópia, reprodução e compartilhamenbtio.

Interleaving

a cada momento, um processo é escolhido para dar um passo



PUCRS – Escola Politécnica – Fernando Luís Dotti - Copyright – direitos reservados. Para uso exclusivo do aluno. Proibida cópia, reprodução e compartilhamenbtio.

Definições

O uso da semântica de *interleaving* para representar o comportamento não implica na execução de fato sequencial de comandos de todos os processos nos processadores.

Mas toda execução do programa concorrente tem uma computação que a descreve.

Vamos ver isto com um exemplo.

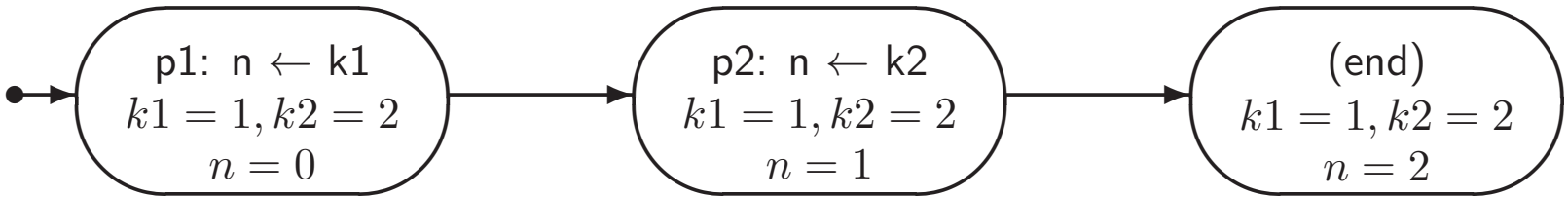
PUCRS – Escola Politécnica – Fernando Luís Dotti - Copyright – direitos reservados. Para uso exclusivo do aluno. Proibida cópia, reprodução e compartilhamenbtio.

Um programa sequencial simples

Algorithm: Trivial sequential program

integer n ← 0

```
      integer k1 ← 1
      integer k2 ← 2
p1:   n ← k1
p2:   n ← k2
```



PUCRS – Escola Politécnica – Fernando Luís Dotti - Copyright – direitos reservados. Para uso exclusivo do aluno. Proibida cópia, reprodução e compartilhamenbtio.

Um programa concorrente simples

Algorithm: Trivial concurrent program	
integer n $\leftarrow$ 0	
p	**q**
integer k1 $\leftarrow$ 1	integer k2 $\leftarrow$ 2
p1: n $\leftarrow$ k1	q1: n $\leftarrow$ k2

PUCRS – Escola Politécnica – Fernando Luís Dotti - Copyright – direitos reservados. Para uso exclusivo do aluno. Proibida cópia, reprodução e compartilhamenbtio.

Representação

Algorithm: Trivial concurrent program	
integer n $\leftarrow$ 0	
p	**q**
integer k1 $\leftarrow$ 1	integer k2 $\leftarrow$ 2
p1: n $\leftarrow$ k1	q1: n $\leftarrow$ k2

```
main(){
   cria e inicia p
   cria e inicia q
   aguarda fim
}
```

PUCRS – Escola Politécnica – Fernando Luís Dotti - Copyright – direitos reservados. Para uso exclusivo do aluno. Proibida cópia, reprodução e compartilhamenbtio.

Representação

Algorithm: Trivial concurrent program	
integer n ← 0	
p	**q**
integer k1 ← 1 p1: n ← k1	integer k2 ← 2 q1: n ← k2

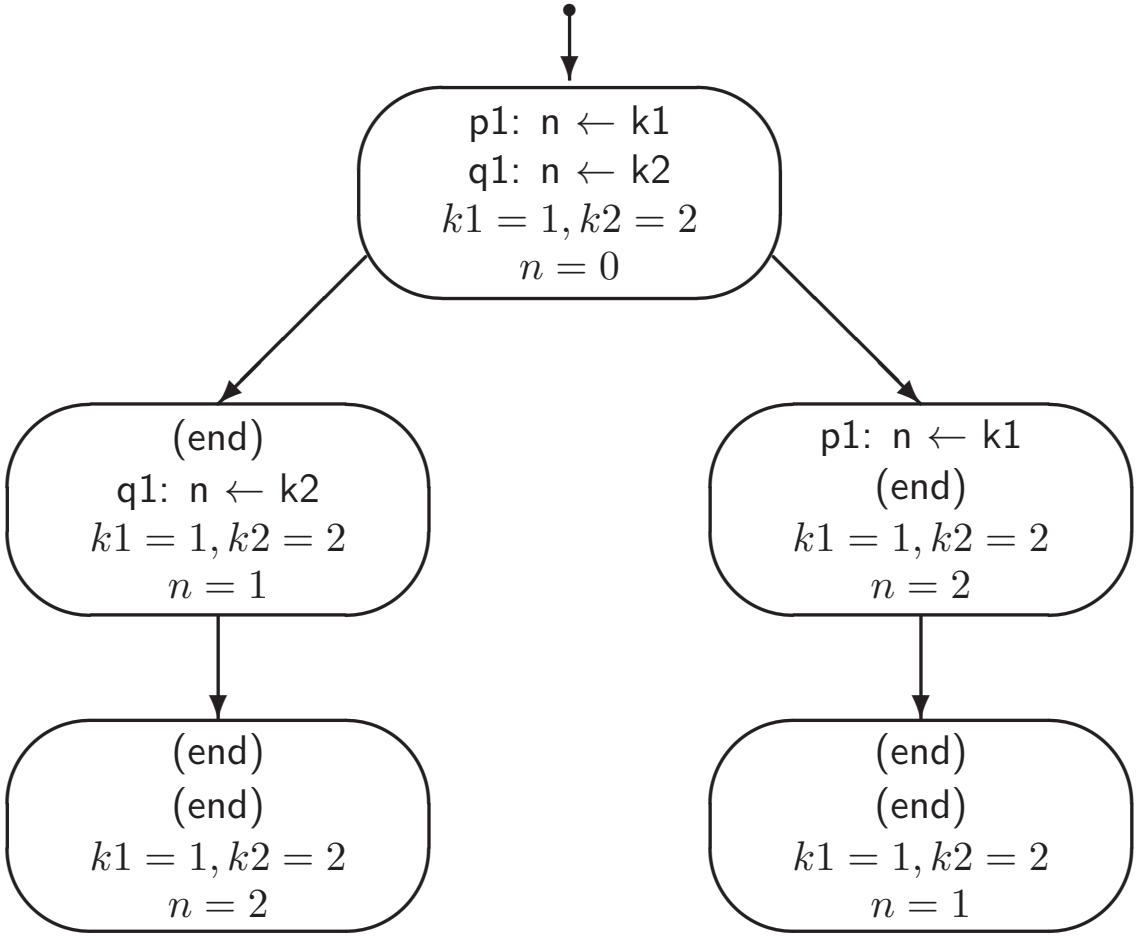
```
main(){
   cria e inicia p
   cria e inicia q
   aguarda fim
}
```

quais os estados e transições ?

PUCRS – Escola Politécnica – Fernando Luís Dotti - Copyright – direitos reservados. Para uso exclusivo do aluno. Proibida cópia, reprodução e compartilhamenbtio.

Um programa concorrente simples

Algorithm: Trivial concurrent program	
integer n ← 0	
p	**q**
integer k1 ← 1	integer k2 ← 2
p1: n ← k1	q1: n ← k2



PUCRS – Escola Politécnica – Fernando Luís Dotti - Copyright – direitos reservados. Para uso exclusivo do aluno. Proibida cópia, reprodução e compartilhamenbtio.

Definições

Um estado de um programa concorrente é uma tupla com:

- um rótulo de cada processo
  - (*program counter* para o processo)
- o valor de cada variável local ou global

PUCRS – Escola Politécnica – Fernando Luís Dotti - Copyright – direitos reservados. Para uso exclusivo do aluno. Proibida cópia, reprodução e compartilhamenbtio.

Definições

Sejam s1 e s2 estados de um programa concorrente, existe uma **transição de s1 para s2** se executando-se um comando em s1 leva a s2.

O comando executado é um dos apontados pelos contadores de programa dos processos no estado s1.

PUCRS – Escola Politécnica – Fernando Luís Dotti - Copyright – direitos reservados. Para uso exclusivo do aluno. Proibida cópia, reprodução e compartilhamenbtio.

Definições

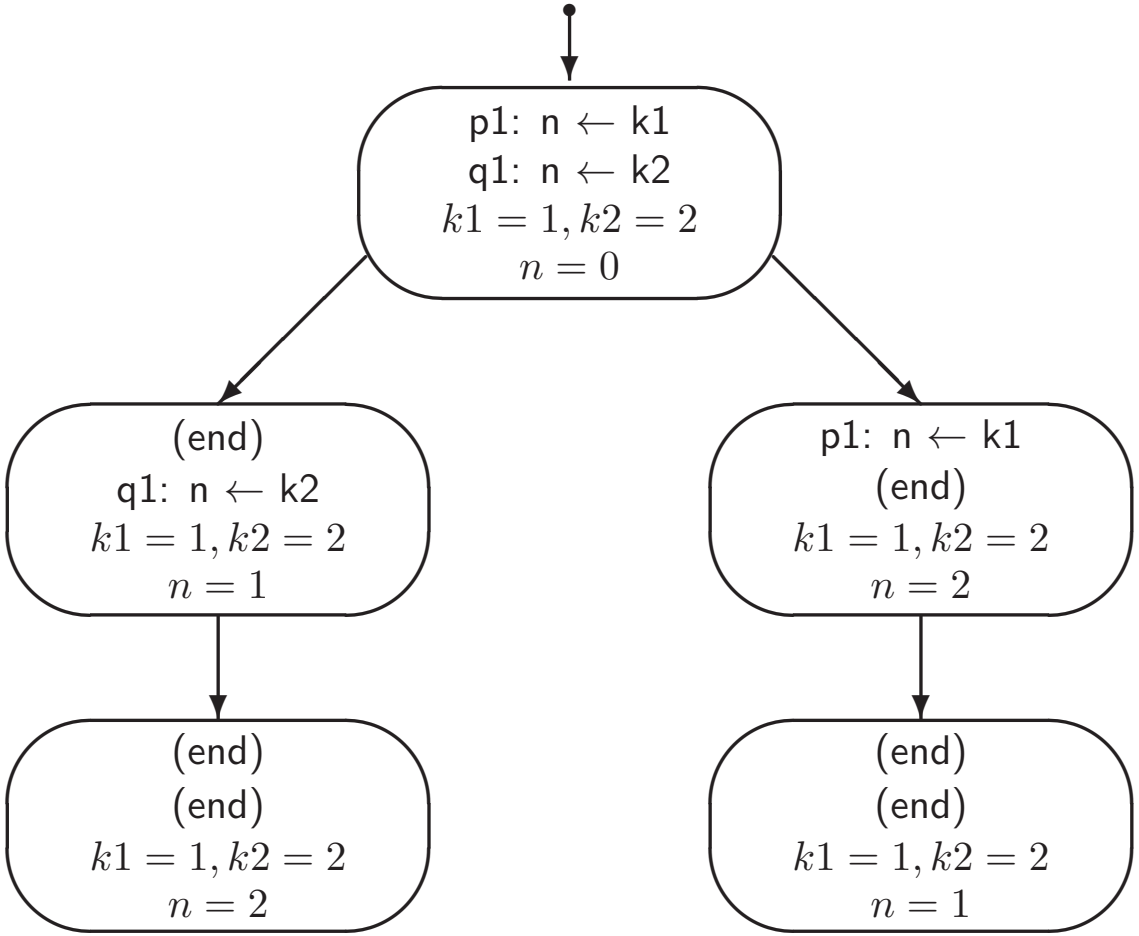
Partindo-se de um estado inicial,
um diagrama de estados é criado indutivamente pela aplicação das transições possíveis.

Este conjunto de estados é dito **conjunto de estados alcançáveis**.

PUCRS – Escola Politécnica – Fernando Luís Dotti - Copyright – direitos reservados. Para uso exclusivo do aluno. Proibida cópia, reprodução e compartilhamenbtio.

Um programa concorrente simples

Algorithm: Trivial concurrent program	
integer n ← 0	
p	**q**
integer k1 ← 1	integer k2 ← 2
p1: n ← k1	q1: n ← k2



PUCRS – Escola Politécnica – Fernando Luís Dotti - Copyright – direitos reservados. Para uso exclusivo do aluno. Proibida cópia, reprodução e compartilhamenbtio.

Definições

Uma computação é **um caminho** dirigido no diagrama de estados,
iniciando no estado inicial.

Ciclos no diagrama representam a possibilidade de computações infinitas.

Estados sem arestas de saída representam situações de **bloqueio** ou **terminação**.

PUCRS – Escola Politécnica – Fernando Luís Dotti - Copyright – direitos reservados. Para uso exclusivo do aluno. Proibida cópia, reprodução e compartilhamenbtio.

Ex.:Concorrência em Go

Go rotinas

- **main** é um processo sequencial
- **go** cria outro processo sequencial executando a função especificada

```
package main

import (
        "fmt"
)

func say(s string) {
        for i := 0; i < 5; i++ {
                fmt.Println(s)
        }
}

func main() {
        go say("world")
        say("hello")
}
```

PUCRS – Escola Politécnica – Fernando Luís Dotti - Copyright – direitos reservados. Para uso exclusivo do aluno. Proibida cópia, reprodução e compartilhamenbtio.

Representação

```
package main
import (
        "fmt"
)
func say(s string) {
        for i := 0; i < 5; i++ {
                fmt.Println(s)
        }
}
func main() {
        go say("world")
        say("hello")
}
```

processo

```
func main() {
        go say("world")
        say("hello")
}
```

processo

```
func say("world") {
        for i := 0; i < 5; i++ {
        fmt.Println("world")
        }
}
```

Tela:

PUCRS – Escola Politécnica – Fernando Luís Dotti - Copyright – direitos reservados. Para uso exclusivo do aluno. Proibida cópia, reprodução e compartilhamenbtio.

Representação

```
package main
import (
        "fmt"
)
func say(s string) {
        for i := 0; i < 5; i++ {
                fmt.Println(s)
        }
}
func main() {
        go say("world")
        say("hello")
}
```

processo

```
func main() {
        go say("world")
        say("hello")
}
```

processo

```
func say("world") {
        for i := 0; i < 5; i++ {
        fmt.Println("world")
        }
}
```

Tela:

world
hello
hello
world
world
hello
hello
world
world
hello

PUCRS – Escola Politécnica – Fernando Luís Dotti - Copyright – direitos reservados. Para uso exclusivo do aluno. Proibida cópia, reprodução e compartilhamenbtio.

Representação

```
package main
import (
        "fmt"
)
func say(s string) {
        for i := 0; i < 5; i++ {
                fmt.Println(s)
        }
}
func main() {
        go say("world")
        say("hello")
}
```

processo

```
func main() {
        go say("world")
        say("hello")
}
```

processo

```
func say("world") {
        for i := 0; i < 5; i++ {
        fmt.Println("world")
        }
}
```

Saída 1 ??:

```
hello
hello
hello
hello
hello
```

Saída 2 ??:

```
hello
hello
hello
hello
hello
world
```

Saída 3 ??:

```
hello
world
world
world
world
world
hello
hello
hello
hello
```

Tela:

world
hello
hello
world
world
world
hello
hello
world
world
hello

PUCRS – Escola Politécnica – Fernando Luís Dotti - Copyright – direitos reservados. Para uso exclusivo do aluno. Proibida cópia, reprodução e compartilhamenbtio.

Representação

```go
package main

import (
    "fmt"
    "time"
)

var N int = 4

func funcaoA(id int, s string) {
    for {
        fmt.Println(s, id)
    }
}

func geraNespacos(n int) string {
    s := "  "
    for j := 0; j < n; j++ {
        s = s + "   "
    }
    return s
}

func main() {
    for i := 0; i < N; i++ {
        go funcaoA(i, geraNespacos(i))
    }
    for true {
        time.Sleep(100 * time.Millisecond)
    }
}
```

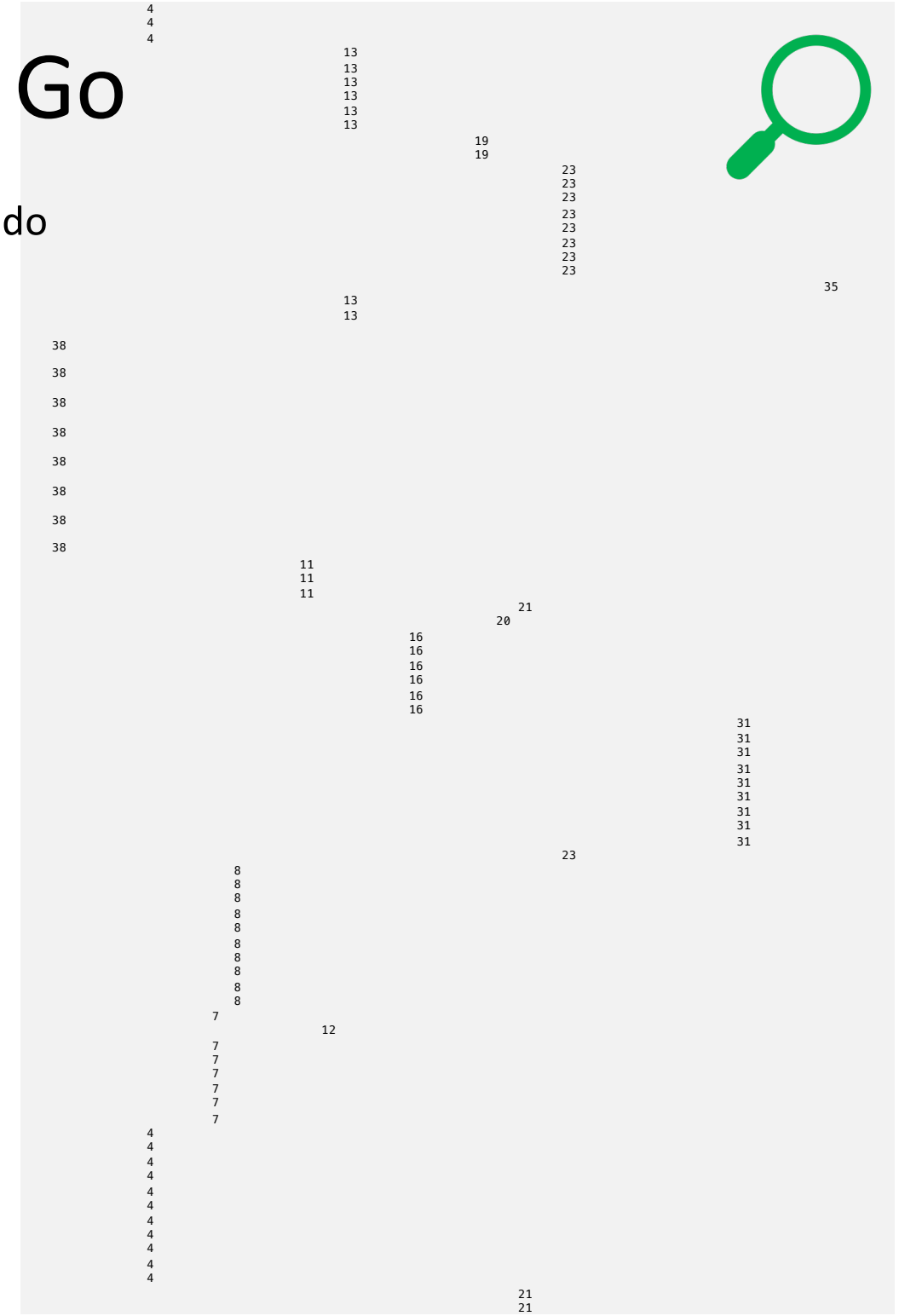
PUCRS – Escola Politécnica – Fernando Luís Dotti - Copyright – direitos reservados. Para uso exclusivo do aluno. Proibida cópia, reprodução e compartilhamenbtio.

Ex.: Concorrência em Go

Go rotinas – velocidades relativas

- veja extrato de saída do programa, ao lado

```go
package main

import (
    "fmt"
    "time"
)

var N int = 4

func funcaoA(id int, s string) {
    for {
        fmt.Println(s, id)
    }
}

func geraNespacos(n int) string {
    s := "  "
    for j := 0; j < n; j++ {
        s = s + "   "
    }
    return s
}

func main() {
    for i := 0; i < N; i++ {
        go funcaoA(i, geraNespacos(i))
    }
    for true {
        time.Sleep(100 * time.Millisecond)
    }
}
```



PUCRS – Escola Politécnica – Fernando Luís Dotti - Copyright – direitos reservados. Para uso exclusivo do aluno. Proibida cópia, reprodução e compartilhamenbtio.

Ex.:Concorrência em Java

Classe Thread

- permite definir um processo leve concorrente
- **método run** define o comportamento da thread
- thread compartilha variáveis conforme regras de escopo da linguagem

```java
/* PUCRS - Programacao Concorrente - Fernando Dotti */

class TesteCriacao extends Thread {
    private int id;
    private int n;
    private String s;

    public TesteCriacao(int _id, int _n, String _s){
      id = _id;
      n = _n;
      s = _s;
    }

    public void run() {
       for (int i = 0; i < n; i++) {
      System.out.println(s + " id: "+id+" -> "+i);
        i++;
      }
    }
}

class Teste1 {
    public static void main(String[] args) {
      TesteCriacao p = new TesteCriacao(1,1000," ");
      TesteCriacao q = new TesteCriacao(2,1000,"                    ");
      p.start();
      q.start();
      try { p.join(); q.join(); }
      catch (InterruptedException e) { }
      System.out.println("Fim");
    }
}
```

PUCRS – Escola Politécnica – Fernando Luís Dotti - Copyright – direitos reservados. Para uso exclusivo do aluno. Proibida cópia, reprodução e compartilhamenbtio.

Ex.:Concorrência em C

com biblioteca PThreads

```c
#include <stdlib.h>
#include <pthread.h>

void *print_message_function( void *ptr );

main()
{
     pthread_t thread1, thread2;
     const char *message1 = "Thread 1";
     const char *message2 = "Thread 2";
     int  iret1, iret2;

    /* Create independent threads each of which will execute function */

     iret1 = pthread_create( &thread1, NULL, print_message_function, (void*) message1);
     if(iret1)
     {
         fprintf(stderr,"Error - pthread_create() return code: %d\n",iret1);
         exit(EXIT_FAILURE);
     }

     iret2 = pthread_create( &thread2, NULL, print_message_function, (void*) message2);
     if(iret2)
     {
         fprintf(stderr,"Error - pthread_create() return code: %d\n",iret2);
         exit(EXIT_FAILURE);
     }

     printf("pthread_create() for thread 1 returns: %d\n",iret1);
     printf("pthread_create() for thread 2 returns: %d\n",iret2);

     /* Wait till threads are complete before main continues. Unless we  */
     /* wait we run the risk of executing an exit which will terminate   */
     /* the process and all threads before the threads have completed.   */

     pthread_join( thread1, NULL);
     pthread_join( thread2, NULL);

     exit(EXIT_SUCCESS);
}

void *print_message_function( void *ptr )
{
     char *message;
     message = (char *) ptr;
     printf("%s \n", message);
}
```

PUCRS – Escola Politécnica – Fernando Luís Dotti - Copyright – direitos reservados. Para uso exclusivo do aluno. Proibida cópia, reprodução e compartilhamenbtio.

Sobre Interleaving Arbitrário

Uma abstração para raciocinar sobre sistemas concorrentes – podemos não ter acesso a um estado global de um sistema concorrente em curso.

Ex.:

- em múltiplos núcleos cada um progride independentemente;
- em sistemas distribuídos cada nodo também;
- e em um sistema monoprocessado o escalonador pode tomar qualquer decisão de qual processo progride em um determinado momento.

PUCRS – Escola Politécnica – Fernando Luís Dotti - Copyright – direitos reservados. Para uso exclusivo do aluno. Proibida cópia, reprodução e compartilhamenbtio.

Sobre Interleaving Arbitrário

Assim, considerar interleaving arbitrário do sistema **retira qualquer suposição temporal** ao ambiente de execução, como velocidade relativa de processos.

PUCRS – Escola Politécnica – Fernando Luís Dotti - Copyright – direitos reservados. Para uso exclusivo do aluno. Proibida cópia, reprodução e compartilhamenbtio.

Justiça - Fairness

Apesar de considerar que *interleaving*, ou seja, qualquer escolha de próxima transição, entre as possíveis, é uma abstração apropriada, temos que fazer uma restrição:

Justiça: não faz sentido supor a possibilidade de os comandos de um processo *nunca* serem selecionados para execução.

PUCRS – Escola Politécnica – Fernando Luís Dotti - Copyright – direitos reservados. Para uso exclusivo do aluno. Proibida cópia, reprodução e compartilhamenbtio.

Justiça - Fairness

Justiça: não faz sentido supor a possibilidade de os comandos de um processo *nunca* serem selecionados para execução.

Uma computação é justa (*weak fairness*) se um comando que está continuamente habilitado nela acaba por ser executado em um momento.

PUCRS – Escola Politécnica – Fernando Luís Dotti - Copyright – direitos reservados. Para uso exclusivo do aluno. Proibida cópia, reprodução e compartilhamenbtio.

Justiça - Fairness

Algorithm: Stop the loop A	
integer n $\leftarrow$ 0 boolean flag $\leftarrow$ false	
p	**q**
p1: while flag $=$ false p2: n $\leftarrow 1 - n$	q1: flag $\leftarrow$ true q2:

Monte o diagrama de estados do algoritmo acima.

Todas as suas computações acabam ?

PUCRS – Escola Politécnica – Fernando Luís Dotti - Copyright – direitos reservados. Para uso exclusivo do aluno. Proibida cópia, reprodução e compartilhamenbtio.

Justiça - Fairness

Algorithm: Stop the loop A	
integer n $\leftarrow$ 0 boolean flag $\leftarrow$ false	
p	**q**
p1: while flag = false p2: n $\leftarrow$ 1 $-$ n	q1: flag $\leftarrow$ true q2:

Existe uma computação não-terminante injusta.

Injusta pois *q está continuamente habilitado*, mas não executa.

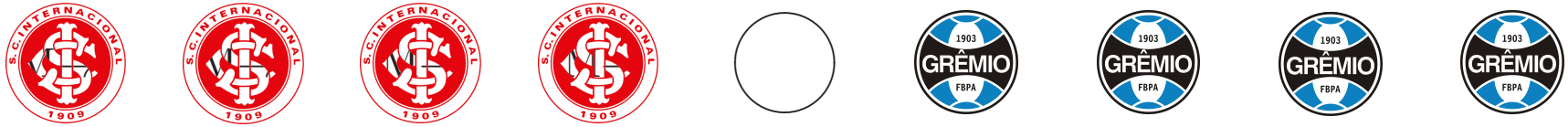
Se restringirmos somente a execuções justas, então q executa e termina, e assim p termina.

Dizemos que com a suposição de weak fairness, o algoritmo termina.

PUCRS – Escola Politécnica – Fernando Luís Dotti - Copyright – direitos reservados. Para uso exclusivo do aluno. Proibida cópia, reprodução e compartilhamenbtio.

Exercício (de Ben-Ari)

Suponha que temos 2n+1 pedras alinhadas em um lago. Nas n pedras da esquerda sentam n sapos colorados e nas n da direita sentam gremistas. Uma pedra no meio está livre. (estado inicial)



Os colorados tentam pular para a direita.

Os gremistas para a esquerda.

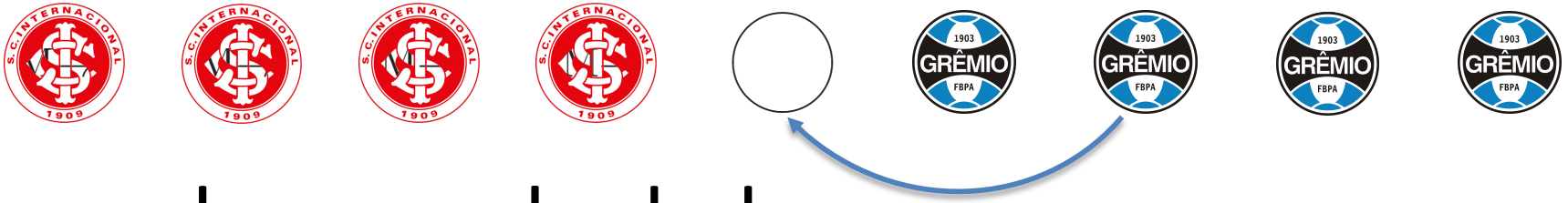
Se uma pedra vizinha está livre:
um sapo pode pular para ela.

Se a pedra posterior à vizinha está livre:
ele pode pular por cima do sapo vizinho.

PUCRS – Escola Politécnica – Fernando Luís Dotti - Copyright – direitos reservados. Para uso exclusivo do aluno. Proibida cópia, reprodução e compartilhamenbtio.

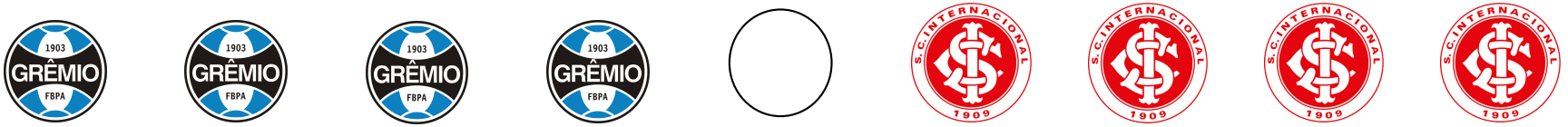
Exercício (de Ben-Ari)

Estado inicial:



Exempo de um pulo dado:

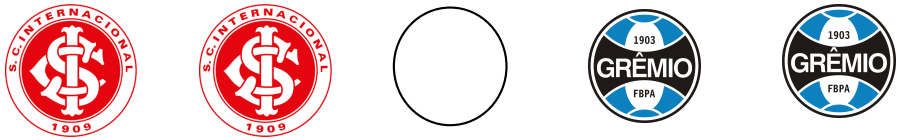
Existe a possibilidade de se atingir este estado?



PUCRS – Escola Politécnica – Fernando Luís Dotti - Copyright – direitos reservados. Para uso exclusivo do aluno. Proibida cópia, reprodução e compartilhamenbtio.

Exercício (de Ben-Ari)

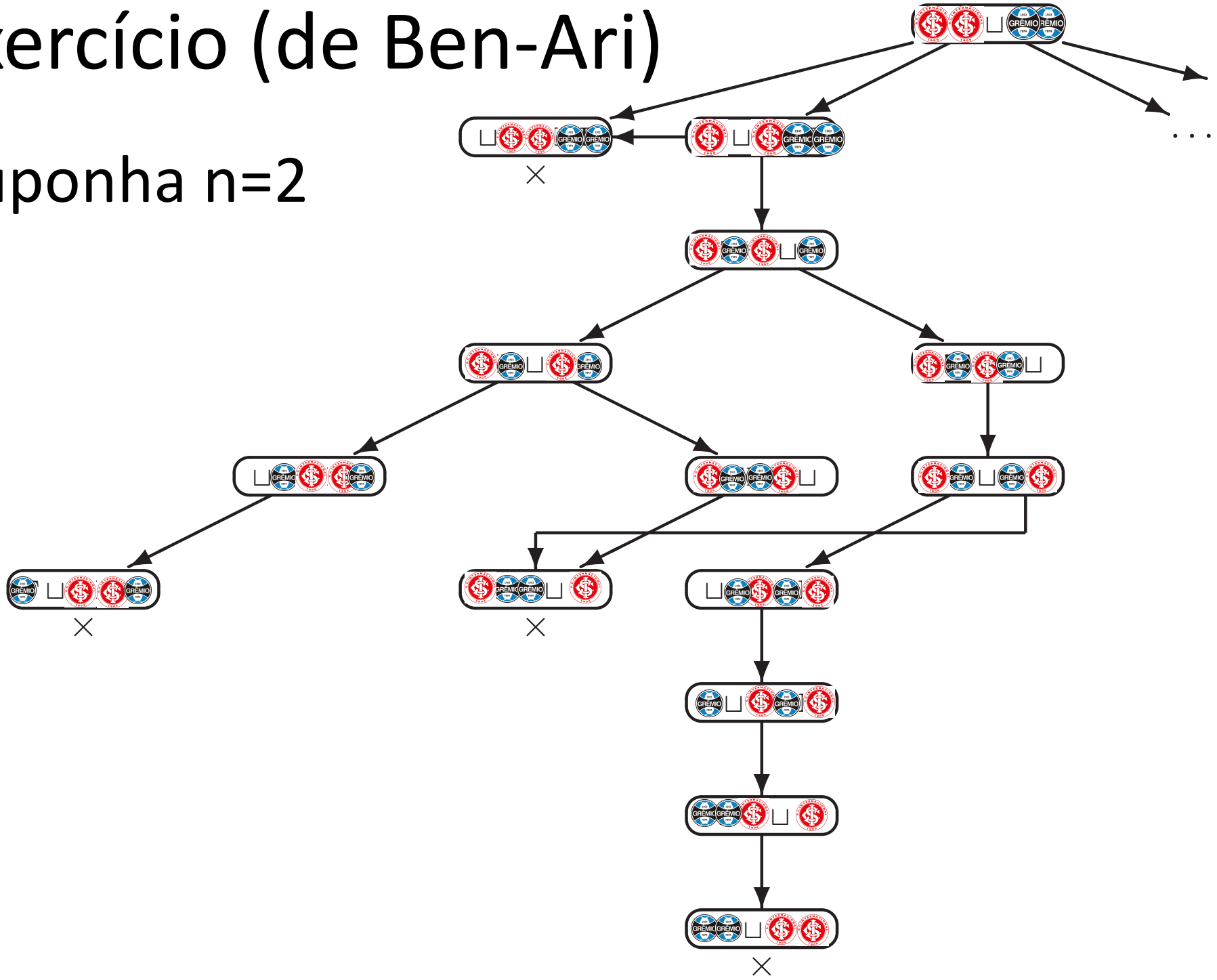
Suponha n=2



PUCRS – Escola Politécnica – Fernando Luís Dotti - Copyright – direitos reservados. Para uso exclusivo do aluno. Proibida cópia, reprodução e compartilhamenbtio.

Exercício (de Ben-Ari)

Suponha n=2



PUCRS – Escola Politécnica – Fernando Luís Dotti - Copyright – direitos reservados. Para uso exclusivo do aluno. Proibida cópia, reprodução e compartilhamenbtio.