Introdução a Algoritmos Distribuídos

Modelo Síncrono

MP003 - Computação Distribuída

Ricardo Anido

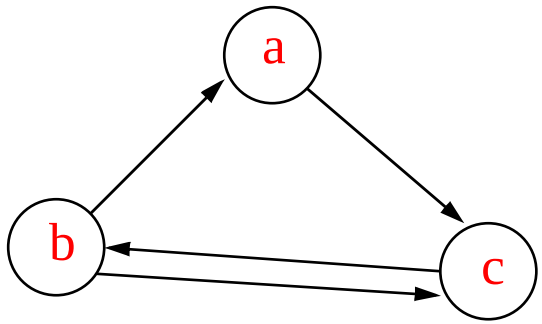
Instituto de Computação

UNICAMP

O Modelo Síncrono

Neste modelo, o sistema é composto de uma coleção de elementos de computação localizados em nós de uma rede

- Rede é modelada como um grafo direcionado G=(V, E).
  - |V| é o número de nós
  - *viz-saída*$_i$ são os vizinhos de saída (arestas que partem de *i* em G)
  - *viz-entrada*$_i$ são os vizinhos de entrada (arestas que chegam a *i* em G)
  - *distância(i, j)* é o comprimento do caminho mínimo entre *i* e *j*.
- Um *processo* é associado com cada vértice em G.



viz-saída$_b$ = {*a*, *c*}

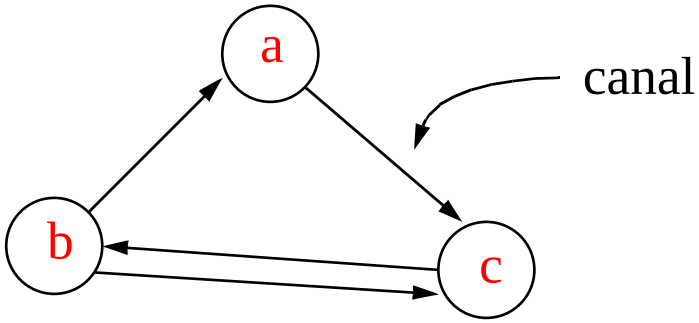
viz-entrada$_b$ = {*c*}

Processos e Canais

Processo é composto por

- variáveis
  - geram seqüências de estados durante a execução
- código
  - controla a seqüências de mensagens enviadas pelo processo

Um *canal* (ou *link*) é associado a cada aresta (*i*, j) de *G*. Um canal é um local que pode armazenar, a cada momento, no máximo uma única mensagem.



Execução

A execução do sistema inicia com os processos em estados iniciais arbitrários, com canais vazios. Processos então executam os seguintes dois passos continuamente, de maneira sincronizada (todos ao mesmo tempo):

1. De acordo com o seu código, gera as mensagens que devem ser enviadas a todos os vizinhos de saída. Coloca as mensagens nos canais apropriados.
2. Retira mensagens dos canais, e atualiza variáveis de acordo com as mensagens recebidas.

Combinação dos dois passos é chamada de um *turno* (*round).*

- Modelo é determinístico.
- Parada de processos: pode-se determinar que em alguns estados não há atividade subsequente (*estados de parada*). Não é o mesmo que estados *finais* em autômatos finitos.

Execução (cont.)

- Pode-se desejar que processos iniciem em tempos distintos:
  - modelo deve ser aumentado com processo de ambiente, conectado com todos os outros processos. Processo de ambiente deve enviar mensagens de *acordar* para todos os outros processos.
  - Processos em estados iniciais não enviam mensagens ou mudam de estado até receberem uma mensagem de *acordar* do processo ambiente ou qualquer outra mensagem não nula de outros processos.
  - Grafos não direcionados: mesmo modelo, com arestas bi-direcionadas.

Falhas

Falhas de processos

- falha-e-pára
  - pára antes ou depois dos passos 1 ou 2.
- omissão
  - pára no meio do passo 1.
- bizantina
  - faz qualquer coisa!

Falhas de canais

- omissão
  - processo produz mensagem no passo 1 mas canal não armazena a mensagem.

Noção de Execução

Para raciocinar sobre o comportamento de um sistema de rede síncrono, é necessária uma noção formal de uma *execução* do sistema.

- Uma *atribuição de estados* de um sistema é definida como uma atribuição de um estado a cada processo no sistema.
- Uma *atribuição de mensagens* é uma atribuição de uma mensagem (possivelmente nula) a cada canal.
- Uma *execução* é uma sequência infinita

$$C_0, M_1, N_1, C_1, M_2, N_2, C_2 \ldots, \text{ onde}$$

C_r é atribuição de estados (C_r representa o estado do sistema após r turnos).

M_r, N_r é atribuição de mensagens (respectivamente mensagens enviadas e recebidas no turno r)

Noção de Execução

Se *a* e *a'* são duas execuções de um sistema, dizemos que *a* é indistiguível de *a'* ($a \sim_i a'$) com relação ao processo *i* se *i* tem a mesma sequência de estados, a mesma sequência de mensagens de saída e a mesma sequência de mensagens de entrada em *a* e *a'*.

Uma visão parcial:

> Se *a* e *a'* são duas execuções de um sistema, dizemos que *a* é indistiguível de *a'* ($a \sim_i a'$) com relação ao processo *i* até o turno *r* se *i* tem a mesma sequência de estados, a mesma sequência de mensagens de saída e a mesma sequência de mensagens de entrada, até o turno *r*, em *a* e *a'*.

Métodos de Prova de Correção

Invariantes:

- Uma invariante é uma propriedade de um estado do sistema (em particular, dos estados de todos os processos) que é verdadeira em cada execução, após cada turno.
- O número de turnos completados pode ser parte da asserção da invariante.
- São provadas por indução no número de turnos, começando com $r = 0$.

Simulação:

- Objetivo é mostrar que algoritmo *A* 'implementa' outro algoritmo *B*, no sentido de produzir o mesmo comportamento de entrada/saída.
- Correspondência entre *A* e *B* é expressa através de uma *relação de simulação* entre estados de *A* e *B*, quando os dois algoritmos iniciam com as mesmas entradas, executam com o mesmo padrão de falhas, pelo mesmo número de turnos.

Medidas de Complexidade

Complexidade de tempo

- número de turnos necessários para que todas as saídas desejadas sejam produzidas, ou até que todos os processos parem.

Complexidade de comunicação

- número total de mensagens não nulas enviadas.
- número de bits usados nas mensagens