Derivação Formal de Algoritmos Distribuídos Utilizando Propriedades de Progresso

Vladimir O. Di Iorio
Departamento de Informática
Universidade Federal de Viçosa
CEP:36570-000 Viçosa - MG
E-mail: vladimir@dpi.ufv.br

Osvaldo S. F. Carvalho
Departamento de Ciência da Computação
Universidade Federal de Minas Gerais
Caixa Postal 702 - CEP:30161-970 Belo Horizonte - MG
E-mail: vado@dcc.ufmg.br

Resumo

Uma estrutura de dados distribuída possui n componentes, uma em cada sítio de um sistema distribuído (nós de uma rede de computadores). A correção de uma estrutura de dados distribuída é freqüentemente expressa através de um predicado que deve ser mantido invariante. Uma técnica formal para derivação de estruturas de dados distribuídas especificada por um invariante é descrita, cuja correção é definida por um predicado global. Utilizando uma rede com dois nós, mostramos que qualquer predicado é equivalente a uma desigualdade sobre um domínio parcialmente ordenado adequadamente escolhido. O modelo é aplicado a um meio de comunicação onde as mensagens podem ser perdidas ou duplicadas, mas a ordem de envio é preservada. São derivados algoritmos para solucionar problemas entre dois nós em um meio como esse, utilizando um formato mínimo para as mensagens trocadas entre os nós.

Abstract

A distributed data structure has n components, one at each site in a distributed system. The correctness of a distributed data structure is often expressed by means of a predicate that should be kept invariant. A formal technique for derivation of distributed data structures specified by an invariant is described, the correctness of which is defined by some (global) predicate. Using a network with only two nodes, we show that every predicate is equivalent to an inequality on a partially ordered domain suitably chosen. The formal model developed is then used with a communication medium where messages can be lost or duplicated, but the delivery order is maintained. Algorithms to solve problems involving two nodes in a communication medium as described above, are derived using a minimum size for the messages exchanged between the nodes.

1 Introdução

Neste artigo consideramos o desenvolvimento de programas concorrentes executados sobre uma rede de computadores com n nós (*programas distribuídos*). Com o objetivo de alcançar resultados mais genéricos, vamos supor que os processos (cada nó da rede) se comunicam apenas através de passagem de mensagens assíncrona, que não compartilham memória e que não possuem um relógio comum. A sincronização será realizada pelas primitivas **send** (envio de mensagem para um canal não limitado que não bloqueia o fluxo de controle do processo emissor) e **receive** (recepção de mensagem que retarda o processo até que o canal não esteja vazio). Não faremos nenhuma suposição sobre as velocidades relativas das mensagens trocadas entre os nós.

Uma estrutura de dados distribuída é constituída por n componentes, cada um localizado em um nó de uma rede. A correção parcial de uma estrutura como essa é muitas vezes expressa pela exigência da invariância de um predicado envolvendo o estado corrente de componentes situados em mais de um sítio. A exclusão mútua [Lam78, CR83, Mae85] nos oferece um exemplo simples de estrutura distribuída, onde o estado de cada sítio é determinado pelo fato de estar usando ou não um recurso compartilhado e a correção parcial exige que nunca se tenham dois ou mais sítios usando esse recurso simultaneamente.

Provas formais de correção [Hoa69, OG76, Kel76, SA86, Sha93] encontram neste tipo de algoritmos um campo de aplicação onde muitas vezes são imprescindíveis. A impossibilidade de exame por um nó da situação global e atrasos arbitrários na transmissão de mensagens via de regra tornam difícil a produção de algoritmos corretos. Neste trabalho damos seguimento às idéias apresentadas em [CDI94], procurando identificar diretivas para tornar a construção de algoritmos distribuídos uma tarefa mais sistemática. Nosso objetivo principal é tornar a prova de correção uma conseqüência direta da concepção do algoritmo.

Em [CDI94] é apresentada uma técnica formal para derivação de estruturas de dados distribuídas, a partir de um predicado global que se deseja invariante. A técnica se restringe a problemas com apenas dois nós e deriva algoritmos com comportamento "browniano", isto é, sem que um sítio possua uma direção preferencial de progresso. Entretanto, é comum que a especificação de um problema inclua propriedades de progresso como "a seqüência de mensagens enviadas é sempre crescente". Essas propriedades podem ser e têm sido usadas para reduzir a informação necessária nas mensagens, como é o caso do protocolo do Alternating Bit [BSW69]. Neste artigo apresentamos um tratamento que, a partir da suposição de não reordenamento na transmissão de mensagens e de propriedades de progresso garantidas para o comportamento dos nós, deriva algoritmos distribuídos corretos e formatos mínimos para mensagens.

Este artigo está organizado da forma descrita a seguir. Na seção 2 são resumidos os pontos que aproveitamos do desenvolvimento introduzido em [CDI94]. Na seção 3 apresentamos o protocolo do bit alternante. As características do meio de comunicação utilizado são examinadas na seção 4 e na seção 5 mostramos um protocolo genérico para a resolução de problemas com essas características. A seção 6 contém as conclusões e discussões sobre trabalhos futuros.

2 Derivação Formal de Estruturas Distribuídas

Esta seção descreve resumidamente os aspectos teóricos introduzidos em [CDI94]. Considere um problema envolvendo estruturas de dados distribuídas com as seguintes características:

- O algoritmo envolve apenas dois nós que se comunicam unicamente através de mensagens, em um meio de comunicação sem falhas, não compartilhando memória e não possuindo um relógio comum; estes nós serão designados *nó 1* e *nó 2*.

- Usaremos e_1 para representar a componente da estrutura distribuída armazenada no nó 1 e e_2 para representar a componente do nó 2.
- A estrutura de dados estará correta se, e somente se, for satisfeito um determinado predicado $\mathbf{P}(e_1, e_2)$.

Para $i = 1, 2$, vamos chamar de E_i o domínio dos valores que a componente do nó i pode assumir (no caso geral, podemos ter $E_1 \neq E_2$). A estrutura de dados será designada por $\mathbf{E}$ e representada por um par ordenado onde cada componente é um elemento de E_i. Podemos então especificar um problema qualquer com essas características simplesmente determinando um predicado $\mathbf{P} : \mathbf{E} \rightarrow \{true, false\}$ que envolva o estado das componentes de cada nó.

Supondo que o predicado inicialmente é válido, devemos exibir um protocolo que o mantenha invariante. Cada nó poderá alterar independentemente o valor de sua componente, ficando a cargo desse protocolo gerenciar as transições válidas. Para isso, vamos supor a existência de um domínio parcialmente ordenado $(\mathcal{D}, \preceq)$ [Gri88], e funções $f_i : E_i \rightarrow \mathcal{D}$, $i = 1, 2$ que mapeiam os valores de e_1 e e_2, de seus respectivos domínios em $\mathcal{D}$. O domínio parcialmente ordenado escolhido deve ser tal que o predicado possa ser reescrito como $\mathbf{P} \equiv f_1(e_1) \preceq f_2(e_2)$. O formato dessas estruturas será discutido mais adiante.

As transições podem ser classificadas como *seguras* ou *perigosas*. O primeiro tipo engloba as transições que nunca poderiam levar o sistema de um estado correto para um incorreto; essas podem ser executadas a qualquer momento. O segundo tipo, por outro lado, engloba as transições que podem acarretar uma violação do predicado $\mathbf{P}$; para serem executadas, essas devem seguir algum protocolo de comunicação.

Supondo um meio em que a transmissão das mensagens é sempre feita sem distorções ou perdas, o seguinte protocolo irá satisfazer os requisitos necessários, para $i = 1, 2$ e $v, v' \in E_i$:

1. Um único token circula entre os dois nós; se o nó i envia o token, ele conterá o valor de $f_i(e_i)$, avaliado no momento da transmissão.
2. O nó 1 só poderá realizar uma transição $(e_1 = v) \rightarrow (e_1 = v')$, onde $f_1(v') \not\preceq f_1(v)$, quando receber o token trazendo um valor u (representando $f_2(e_2)$) tal que $f_1(v') \preceq u$.
3. O nó 2 só poderá realizar uma transição $(e_2 = v) \rightarrow (e_2 = v')$, onde $f_2(v) \not\preceq f_2(v')$, quando receber o token trazendo um valor u (representando $f_1(e_1)$) tal que $u \preceq f_2(v')$.

Esse protocolo será designado *protocolo do token circulante*. Omitiremos aqui a prova de sua correção, que pode ser facilmente desenvolvida.

O tráfego de mensagens pode ser reduzido consideravelmente se for utilizada uma técnica simples para "distribuição" do predicado $\mathbf{P}$. Para isso, vamos introduzir duas novas variáveis: c_1, no nó 1, e c_2, no nó 2, com ambas assumindo valores dentro do domínio parcialmente ordenado $(\mathcal{D}, \preceq)$. Vamos também substituir o predicado $\mathbf{P} \equiv f_1(e_1) \preceq f_2(e_2)$ pela conjunção dos predicados $L_1 \equiv f_1(e_1) \preceq c_1$, $L_2 \equiv c_2 \preceq f_2(e_2)$, $G \equiv c_1 \preceq c_2$. Os predicados L_1 e L_2 são chamados de *predicados locais* e G é chamado de *predicado global*.

Para mantermos a validade do predicado G, podemos utilizar novamente o protocolo do token circulante. Para o caso dos predicados locais, basta introduzir, sempre que um nó for executar uma transição, testes que garantam que as variáveis locais e_1 e e_2 não irão violar esses predicados. Usando essa abordagem, os nós não precisam necessariamente esperar a chegada de mensagens sempre que forem alterar o valor de sua componente, mesmo no caso de algumas transições perigosas. O ganho vem da memorização do conhecimento adquirido, registrado nas variáveis c_1 e c_2.

2.1 Definição dos Mapeamentos

Considere dois domínios arbitrários E_1 e E_2 para as variáveis locais e_1 e e_2 e um predicado $\mathbf{P}: E_1 \times E_2 \rightarrow \{true, false\}$. É preciso identificar um domínio parcialmente ordenado $(\mathcal{D}, \preceq)$ e funções f_1 e f_2 que mapeiam os domínios E_1 e E_2 e o predicado $\mathbf{P}$ em uma desigualdade em $\mathcal{D}$, de modo que o predicado possa ser reescrito como $\mathbf{P} \equiv f_1(e_1) \preceq f_2(e_2)$.

Como proposta inicial, vamos tomar o conjunto 2^{E_1} dos subconjuntos de E_1, parcialmente ordenado pela relação de inclusão. Precisamos encontrar funções $f_1: E_1 \rightarrow 2^{E_1}$ e $f_2: E_2 \rightarrow 2^{E_1}$ tais que $f_1(e_1) \preceq f_2(e_2) \Leftrightarrow \mathbf{P}(e_1, e_2)$. Para simplificar, vamos fazer $f_1(e_1) = \{e_1\}$. A função f_2 deve funcionar de maneira que $\{e_1\} \preceq f_2(e_2) \Leftrightarrow \mathbf{P}(e_1, e_2)$. Nesse ponto, é conveniente introduzirmos a seguinte definição:

Definição 1 Dados dois conjuntos E_1 e E_2 e um predicado $\mathbf{P}$ sobre $E_1 \times E_2$, dizemos que dois subconjuntos $S_1 \subseteq E_1$ e $S_2 \subseteq E_2$ são $\mathbf{P}$-*compatíveis* se, e somente se, para $e_1 \in E_1$, $e_2 \in E_2$, temos $(e_1, e_2) \in S_1 \times S_2 \Rightarrow \mathbf{P}(e_1, e_2)$. Ou seja, dois conjuntos S_1, S_2 são $\mathbf{P}$-compatíveis se, e somente se, cada par ordenado formado por um elemento de S_1 e um elemento de S_2 satisfizer ao predicado $\mathbf{P}$. □

Vamos definir também o seguinte mapeamento, para $i, j \in \{1, 2\}$, $i \neq j$:

$$\begin{aligned} \mathbf{P}_{ji}(S_j) &= \{e_i \in E_i \mid \forall\, e_j \in S_j, \mathbf{P}(e_i; e_j)\}, \text{ se } S_j \neq \emptyset_j, \\ \mathbf{P}_{ji}(\emptyset_j) &= E_i \end{aligned}$$

onde $(e_i; e_j)$ denota (e_i, e_j) se $i < j$, e (e_j, e_i) se $j < i$. $\mathbf{P}_{ji}$ mapeia um subconjunto S_j de E_j em um subconjunto $\mathbf{P}$-compatível de E_i. Chamaremos o mapeamento $\mathbf{P}_{ji}(S_j)$ de *transformada* de S_j por $\mathbf{P}$.

Podemos adotar então o seguinte formato para f_2: $f_2(e_2) = \mathbf{P}_{21}(\{e_2\})$.

Como $\{e_1\} \subseteq \mathbf{P}_{21}(\{e_2\}) \Leftrightarrow \mathbf{P}(e_1, e_2)$, chegamos ao resultado desejado. Ou seja, dados dois domínios arbitrários E_1 e E_2 e um predicado $\mathbf{P}$, encontramos um domínio parcialmente ordenado $(2^{E_1}, \subseteq)$ e duas funções f_1 e f_2 que mapeiam o predicado original em uma desigualdade sobre esse domínio parcialmente ordenado.

Como vimos anteriormente, podemos utilizar protocolos simples para manter invariante essa desigualdade. Além disso, ela pode ser decomposta em dois predicados locais e um predicado global, seguindo a idéia de economia no tráfego de mensagens que discutimos anteriormente. Assim teremos: $L_1 \equiv \{e_1\} \subseteq c_1$, $L_2 \equiv c_2 \subseteq \mathbf{P}_{21}(\{e_2\})$, $G \equiv c_1 \subseteq c_2$. Nesse caso, chamaremos as variáveis c_1 e c_2 de *territórios de possíveis estados* (ou simplesmente *territórios*) dos nós 1 e 2, respectivamente.

Nesse ponto, algumas questões devem ser levantadas. Nós elegemos 2^{E_1} para ser o domínio parcialmente ordenado com que iríamos trabalhar. Se 2^{E_2} fosse um domínio diferente do primeiro, quais seriam as implicações de escolher este ou aquele? E se o tamanho dos domínios fosse muito discrepante?

Outra questão importante é a seguinte: a estrutura de dados distribuída que utilizamos nessa solução é a mais eficiente? Da forma como especificamos o protocolo do token circulante, a qualquer momento um nó pode realizar uma perda voluntária de seu território, de modo a proporcionar uma possível expansão do território do outro nó. Acontece que não há nenhuma garantia de que essa perda vá permitir a desejada expansão. Isso ficará mais claro no exemplo exibido na seção 2.2.

A teoria introduzida em [CDI94] identifica exatamente aqueles subconjuntos de domínios arbitrários E_1, E_2 cuja representação é relevante para um protocolo que se proponha a manter invariante um predicado $\mathbf{P}$. Essa teoria é baseada na estrutura conhecida como *Conexão de Galois* [Ore44, Sza63]. Se $S_i \subseteq E_i$, a transformada dupla de $\mathbf{P}$ sobre S_i é caracterizada como

uma *operação de fechamento* nos reticulados 2^{E_i}, $i = 1, 2$:

$$\begin{aligned} S_i &\subseteq \mathbf{P}_{ji}(\mathbf{P}_{ij}(Si)) \\ \mathbf{P}_{ji}(\mathbf{P}_{ij}(S_i)) &= \mathbf{P}_{ji}(\mathbf{P}_{ij}(\mathbf{P}_{ji}(\mathbf{P}_{ij}(S_i)))) \end{aligned}$$

Isso leva à importante definição a seguir.

Definição 2 Um subconjunto S_i de E_i é **P**-*fechado* se, e somente se, $S_i = \mathbf{P}_{ji}(\mathbf{P}_{ij}(S_i))$. □

Os **conjuntos P-fechados** são exatamente aqueles que procuramos. Além disso, outros resultados mostram que existe um isomorfismo dual entre o reticulado de subconjuntos **P**-fechados de E_1 e E_2, esclarecendo que não importará qual dos dois domínios será o escolhido na elaboração do protocolo.

2.2 Exemplo

Vamos aplicar as idéias apresentadas nas seções anteriores para produzir um algoritmo que resolva o problema da exclusão mútua envolvendo dois nós. O problema pode ser assim especificado:

$$E_1 = E_2 = \{idle, busy\}$$

$$\mathbf{P} \equiv (e_1 = idle) \vee (e_2 = idle)$$

onde *idle* representa um estado em que o nó não está utilizando o recurso compartilhado e *busy* representa o contrário. O predicado exige que no máximo um dos nós esteja no estado *busy*. Os conjuntos **P**-fechados relacionados são:

$$\begin{aligned} \{idle\} &= \mathbf{P}_{ij}(\mathbf{P}_{ji}(\{idle\})) \\ \{idle, busy\} &= \mathbf{P}_{ij}(\mathbf{P}_{ji}(\{idle, busy\})) \end{aligned}$$

A figura 1 mostra o subconjunto 2^{E_i} ($E_1 = E_2$, nesse caso), utilizando uma representação de conjuntos parcialmente ordenados conhecida como *Diagrama de Hasse*. Na figura 2 pode-se ver os domínios dos subconjuntos **P**-fechados de E_1 e E_2, ressaltando o isomorfismo dual que existe entre eles, onde as correspondências são dadas pelas transformadas $\mathbf{P}_{12}$ e $\mathbf{P}_{21}$.

O algoritmo para resolver esse problema pode ser visto na figura 3. A notação utilizada é a sugerida em [Hoa78], usando os *comandos guardados* introduzidos em [Dij75]. Neste trabalho utilizaremos constantemente essa notação nos algoritmos apresentados. Convém salientar, entretanto, que a comunicação não é feita de maneira síncrona como na linguagem CSP. Estaremos sempre considerando uma comunicação assíncrona [AS83, BST89], sem estabelecer nenhuma suposição sobre a velocidade relativa das mensagens trocadas.

O algoritmo representa uma navegação sobre o domínio de conjuntos **P**-fechados relacionado ao problema. Uma expansão de território só é permitida ao nó i após a recepção de uma mensagem proveniente de j indicando perda voluntária de território. Por outro lado, essa perda voluntária pode ser executada a qualquer momento. Observe como seria indesejável o comportamento desse algoritmo se o protocolo se baseasse no domínio da figura 1: um nó que estivesse com o território corrente $\{idle, busy\}$ poderia restringi-lo para $\{busy\}$. Isso não permitiria uma possível expansão para o outro nó, uma vez que $\mathbf{P}_{ij}(\{busy\}) \hat{=} \{idle\} = \mathbf{P}_{ij}(\{idle, busy\})$. A formalização da derivação de estruturas que contornam esse problema é um dos principais resultados obtidos.

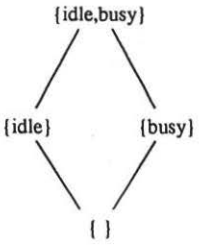

Figura 1: Reticulado de 2^{E_i}, para o problema da exclusão mútua.

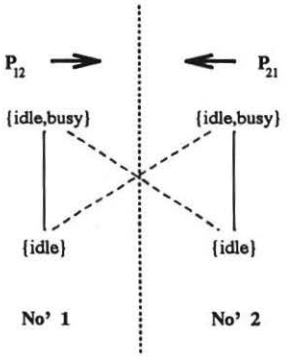

Figura 2: Subconjuntos **P**-fechados para o problema da exclusão mútua.

```
TYPE State = (idle, busy);
  Knowledge = SET OF State;
[ Pi:: e: State; c: Knowledge; hastoken: BOOLEAN;
  e := idle; c := [idle]; hastoken := (i = 1);
  *[TRUE → e := idle;
  | c = [idle,busy] → e := busy;
  | e = idle → c := [idle];
  | receive(token) → c := Pji(token.c);
                hastoken := TRUE;
  | hastoken → token.c := c;
                send(token); hastoken := FALSE;
  ]
]
```

Figura 3: Algoritmo para Exclusão Mútua usando conjuntos **P**-fechados.

3 O Protocolo do Alternating Bit

Os aspectos teóricos exibidos na seção 2 proporcionam a derivação formal de estruturas distribuídas e algoritmos que resolvam problemas envolvendo a comunicação entre dois nós. O protocolo do token circulante é adequado às características de um meio de comunicação que não permite falhas na troca de mensagens.

É comum termos aplicações distribuídas rodando em um meio de comunicação onde as mensagens enviadas podem ser perdidas ou duplicadas, mas cuja ordem de envio é preservada. Um dos problemas mais importantes envolvendo um meio como esse é o seguinte:

1. Temos dois nós que se comunicam através de mensagens, sendo que um é designado *emissor*, e o outro, *receptor*.

2. O predicado que se deseja manter invariante exige que a seqüência de mensagens enviada pelo emissor deve ser idêntica à recebida pelo receptor, ou diferir apenas da última mensagem enviada, a qual pode ainda não ter sido recebida.

Ou seja, o emissor só pode enviar uma nova mensagem quando tiver certeza de que a anterior foi recebida corretamente. A solução clássica para esse problema é dada pelo *Protocolo do Alternating Bit* [BSW69]:

1. Cada mensagem transmitida pelo emissor é acrescida de um bit adicional, que alterna os valores 0 e 1 a cada nova transmissão.

2. O receptor envia, como confirmação do recebimento, a mesma mensagem que chegou (na realidade, basta transmitir o bit adicional).

Muitas provas foram apresentadas para mostrar que esse protocolo funciona adequadamente. Entre elas podemos citar [SMS82].

No protocolo descrito acima, pode-se ver que as características do meio de comunicação possibilitaram a utilização de um formato bastante econômico para as mensagens trocadas entre os nós (apenas um bit adicional é necessário para a recuperação de possíveis perdas ou duplicações). Neste trabalho, vamos propor um modelo formal para solucionar problemas como esse. Nosso objetivo não se resume a apenas apresentar uma prova de correção do *Protocolo AB*. O que se deseja é exibir uma técnica que permita derivar formalmente um protocolo e formato mínimo para as mensagens trocadas, possibilitando resolver qualquer problema envolvendo dois nós e um meio de comunicação como o descrito no início desta seção.

4 Generalização

Nesta seção vamos estudar as particularidades que permitiram ao Protocolo do Bit Alternante simplificar o formato das mensagens trocadas entre os nós. Assim, podemos generalizar os resultados e utilizá-los em outras situações. Seguindo as idéias introduzidas na seção 2, vamos buscar a definição de um protocolo genérico que resolva qualquer problema com as características enumeradas na seção 3. O problema deve ser especificado por um predicado **P** sobre as componentes da estrutura distribuída. Vamos mostrar que os reticulados de subconjuntos **P**-fechados definem exatamente o formato mínimo para as mensagens trocadas entre os nós.

4.1 Conjuntos P-fechados associados

Considere um meio de comunicação que se comporta da maneira como a descrita na seção 3. Vamos aplicar a teoria introduzida na seção 2 ao problema resolvido pelo Protocolo AB. Para

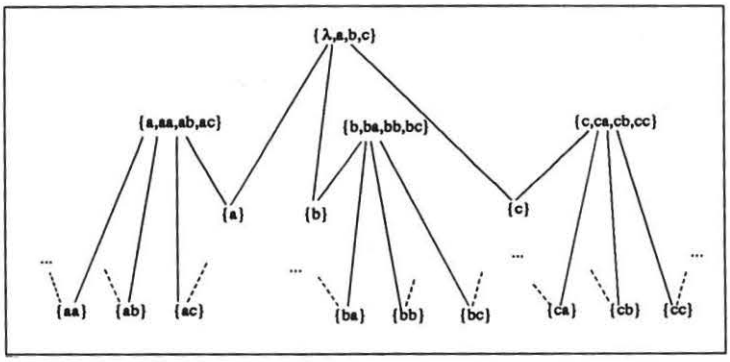

Figura 4: Subconjuntos **P**-fechados do nó 1.

um alfabeto Σ de mensagens, $\sigma \in \Sigma$, $e_1 \in E_1$, $e_2 \in E_2$, o problema pode ser assim especificado:

$$\begin{aligned} \Sigma^* &= E_1 = E_2 \\ \mathbf{P} &\equiv (e_1 = e_2) \vee (e_1 = e_2 \cdot \sigma) \end{aligned}$$

O nó 1 é designado como *emissor* e o nó 2, como *receptor*. A seqüência de símbolos transmitida pelo emissor deve ser igual à recebida pelo receptor, ou diferir apenas no último símbolo, enviado mas ainda não recebido. Como dissemos antes, o meio de transmissão admite perda ou duplicação de mensagens, mas a ordem de envio é preservada. Vamos considerar novamente uma comunicação assíncrona onde atrasos de transmissão podem ser arbitrariamente longos.

Para simplificar, mas sem perda de generalidade, vamos trabalhar com um alfabeto de mensagens $\Sigma = \{a, b, c\}$. Deixando de fora os subconjuntos vazios, podemos ver nas figuras 4 e 5 os domínios parcialmente ordenados formados pelos subconjuntos **P**-fechados de E_1 e E_2, respectivamente. No nó 1, esses subconjuntos são da forma $\{x\}$ e $\{x, x \cdot a, x \cdot b, x \cdot c\}$, para $x \in \Sigma^+$. No nó 2, os subconjuntos **P**-fechados têm o formato $\{y\}$ e $\{y \cdot \sigma\}$, para $y \in \Sigma^*$, $\sigma \in \Sigma$.

Observe o formato do domínio de subconjuntos **P**-fechados de E_2 (figura 5). Uma perda voluntária de território por parte desse nodo sempre resulta na escolha de uma de duas possibilidades, independente do tamanho do alfabeto Σ. Vamos mostrar um protocolo que garante que a atribuição alternada de 0 e 1 aos elementos não maximais possibilita a identificação dos mesmos de forma não-ambígua. O bit alternado usado pelo Protocolo AB poderia ser entendido como a representação dessas duas situações diferentes.

Nas seções seguintes, vamos mostrar que essa idéia pode ser estendida a qualquer problema especificado por um predicado que atua em um meio com as características exigidas. Será exibida uma técnica que permite derivar o formato mínimo para as mensagens trocadas entre os nós, da mesma forma que o Protocolo AB propõe um bit alternado para as mensagens do receptor. Um resultado muito interessante é que a mesma idéia será aplicada também ao emissor, determinando formalmente um formato mínimo para as mensagens enviadas por este.

4.2 Solução Visualizada

Não podemos utilizar um protocolo como o do token circulante para desenvolver um algoritmo distribuído que trate o problema da seção anterior. Obviamente, a perda de uma mensagem acarretaria uma parada no sistema. Suponha que um determinado intervalo de tempo transcorra

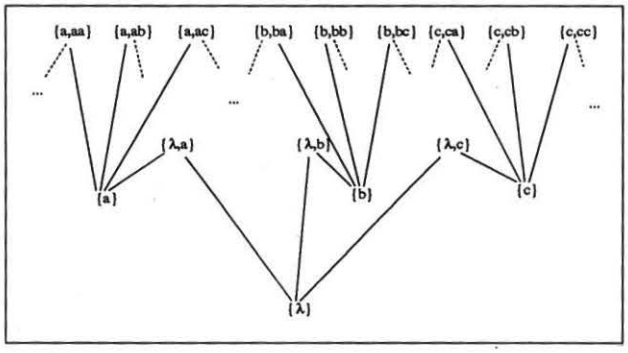

Figura 5: Subconjuntos **P**-fechados do nó 2.

após a última mensagem ter sido enviada por um nó, sem que o mesmo receba uma confirmação da chegada da mesma ao outro nó. Isso *pode* significar que a mensagem foi perdida. Um protocolo que funcione nesse caso deve fazer com que um nó envie repetidamente a mesma mensagem, esperando um certo intervalo de tempo que considere suficiente para que possa receber a confirmação a que nos referimos. A confirmação de uma mensagem recebida pelo receptor pode ser realizada, como no Protocolo AB, através da transmissão do bit adicional. Nos interessa descobrir quais as características inerentes a esse problema que permitiram ser possível a utilização de apenas um bit. Dessa forma, podemos generalizar esse conceito e aplicá-lo em outras situações.

Suponha que o predicado **P**, como descrito na seção 4.1, seja distribuído em predicados locais L_1 e L_2 e um predicado global G. O território de possíveis estados para o nó 1 será designado por c_1 e para o nó 2, por c_2. Sabemos que a representação é equivalente para qualquer um dos dois domínios de subconjuntos **P**-fechados. Para facilitar o estudo dos princípios que desejamos descobrir, será proposta a seguinte distribuição do predicado **P**:

$$\begin{aligned} L_1 &\equiv e_1 \subseteq c_1 \\ L_2 &\equiv e_2 \subseteq c_2 \\ G &\equiv c_1 \subseteq \mathbf{P}_{21}(c_2) \end{aligned}$$

Observando as figuras 4 e 5, vamos supor uma situação onde o território corrente do nó 1 seja um valor que permita ao nó 2 se situar em um maior número de estados diferentes possível. Claramente, o valor a que nos referimos deve ser um elemento minimal do domínio dos subconjuntos **P**-fechados de E_1. Se o predicado for mantido invariante, c_2 poderá ser um elemento maximal do domínio dos subconjuntos **P**-fechados de E_2. Por exemplo, pode-se ter $c_1 = \{a\}$ e assim $c_2 \subseteq \mathbf{P}_{12}(c_1) = \{\Lambda, a\}$.

Vamos considerar essa situação do ponto de vista do nó 1. Suponha que este nó receba uma mensagem que permita ao mesmo identificar o território do nó 2, amostrado no instante do envio. Se o predicado é mantido invariante, o nó 1 pode inferir com certeza que o valor de c_2 é um dos conjuntos a seguir:

$\{\Lambda\}$: pode ser que o nó 2 ainda não tenha recebido a última mensagem enviada pelo nó 1;

$\{\Lambda, a\}$: o nó 2 recebeu essa mensagem e o valor de c_2 passou a ser $\mathbf{P}_{21}(\{a\}) = \{\Lambda, a\}$;

$\{a\}$: o nó 2 restringiu voluntariamente o seu território após tê-lo expandido, para permitir em seguida uma expansão do território do nó 1.

Dos três casos que acabamos de listar, o segundo tem uma particularidade interessante. Podemos afirmar que não interessa, no protocolo que pretendemos desenvolver, que o nó 2 envie uma mensagem indicando o valor de c_2 para esse caso. Isso acontece porque, sendo $\{\Lambda, a\}$ um elemento maximal, certamente não possibilitará ao nó 1 nenhum tipo de expansão. Ou seja, o valor de c_1 é, com certeza, igual a $\{a\}$ e não poderá sofrer nenhuma alteração mesmo que receba essa mensagem.

Assim, chegamos a um ponto onde o nó 1 sabe que, se o protocolo de comunicação funciona corretamente e o predicado é mantido invariante, então apenas duas mensagens diferentes poderão ser recebidas por ele. Uma pode representar que o nó 2 ainda não recebeu a mensagem que indica $c_1 = \{a\}$, e a outra pode representar a perda voluntária de território do nó 2, após receber essa mensagem. Um ponto que devemos ressaltar é que, para chegar a essa conclusão, o nó 1 não precisa apenas saber que o seu território corrente é **P**-compatível com o do nó 2. Ele deve também assumir a premissa de que, se recebe uma mensagem representando o valor do território do nó 2 em um momento no passado, esse valor é **P**-compatível com o seu território atual. Ou seja, a **P**-compatibilidade seria obedecida também na "diagonal". Este é um importante resultado utilizado neste trabalho, cuja demonstração é realizada formalmente em [DI94].

4.3 Compromissos de Progresso

Um outro aspecto deve ser considerado, antes de podermos derivar um algoritmo distribuído completo. Suponha uma situação onde, para $i, j \in \{1, 2\}$, $i \neq j$, o nó i recebe uma mensagem representando o território do nó j, amostrado no instante do envio da mensagem. Seguindo as idéias introduzidas na seção 2, o nó i ganharia o direito de realizar uma expansão de seu território que fosse **P**-compatível com o valor representado na mensagem recebida. Essa regra não poderá ser seguida de forma irrestrita nesse novo meio de comunicação. O nó i deverá considerar também a possível expansão que o território do nó j pode ter sofrido desde que enviou essa mensagem, como veremos a seguir.

Vamos supor, por exemplo, que o nó 1 enviou uma mensagem ao nó 2, representando $c_1 = \{a\}$. Como havíamos concluído no início desta seção, ele pode resolver repetir a transmissão da mesma, considerando uma possível falha na comunicação. Ao receber a primeira dessas duas mensagens, o nó 2 pode fazer $c_2 = \mathbf{P}_{12}(\{a\}) = \{\Lambda, a\}$, executar em seguida uma perda voluntária de território e enviar uma mensagem representando o novo território, que seria $\{a\}$. Vamos supor que, num dado momento no futuro, o nó 1 receba essa mensagem e faça então $c_1 = \mathbf{P}_{21}(\{a\}) = \{a, aa, ab, ac\}$, mas a segunda cópia de $c_1 = \{a\}$ chegue ao nó 2, que resolva fazer novamente $c_2 = \{\Lambda, a\}$. Teríamos então

$$c_2 = \{\Lambda, a\} \not\subseteq \{a\} = \mathbf{P}_{12}(\{a, aa, ab, ac\}) = \mathbf{P}_{12}(c_1).$$

Isso caracterizaria uma violação do predicado global G.

Um fato que estamos desconsiderando e que é aproveitado pelo Protocolo AB, é o seguinte: uma vez que notificou o recebimento de uma certa seqüência de mensagens, o receptor não pode "recuar" dessa decisão. Ou seja, o valor de c_2 não pode passar de $\{a\}$ para $\{\Lambda, a\}$, mesmo que receba uma mensagem do nó 1 indicando território **P**-compatível com $\{\Lambda, a\}$. De forma semelhante, uma vez que o emissor envia mais um símbolo para compor a seqüência, não lhe é permitido desistir desse símbolo e enviar um novo em seu lugar. Esses fatos são características inerentes a este problema, constituindo um conhecimento que não pode ser extraído apenas do predicado **P**.

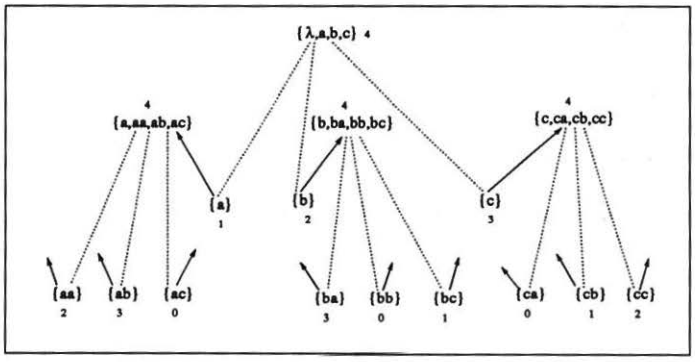

Figura 6: Compromisso de Progresso do nó 1.

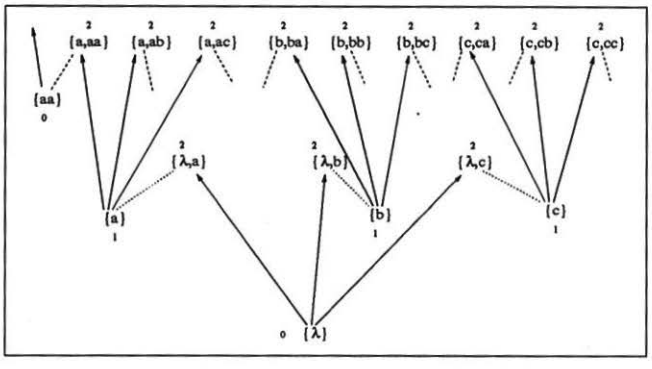

Figura 7: Compromisso de Progresso do nó 2.

Esses conceitos são formalizados em [DI94] por uma estrutura designada por *Compromissos de Progresso*. Essas estruturas caracterizam um "acordo" obedecido por ambos os nós, de modo a disciplinar as alterações permitidas em sua componente da estrutura de dados distribuída. A representação do acordo é feita através de um grafo direcionado onde os vértices são os subconjuntos **P**-fechados associados. As direções das arestas representam restrições que os nós se comprometem a cumprir na expansão dos seus territórios quando da recepção de mensagens.

Para o problema em questão, pode-se ver esse acordo representado pelos Compromissos de Progresso das figuras 6 e 7 – os valores numéricos associados a cada subconjunto são discutidos mais adiante. Por exemplo, observando a figura 6, se o território corrente do nó 1 for $c_1 = \{ab\}$ (enviou como segunda mensagem o símbolo b e espera resposta) e receber a mensagem representando $c_2 = \{a\}$ (recepção do primeiro símbolo), o nó 1 não pode expandir seu território para $\mathbf{P}_{21}(\{a\}) = \{a, aa, ab, ac\}$. As arestas impõem uma direção para o progresso das transições.

Assim, a especificação completa de um problema, para que as técnicas propostas possam ser usadas, deve ser composta de:

1. um predicado **P** sobre $\mathbf{E} = E_1 \times E_2$, e
2. um conjunto de restrições sobre as possíveis transições a serem realizadas, que pode ser interpretado como um "acordo" entre os dois nós.

5 Protocolo Proposto

Seguindo as idéias discutidas na seção anterior, o protocolo genérico de comunicação entre dois nós que iremos propor terá um formato como o descrito abaixo:

- Cada nó irá manter uma variável representando o seu território corrente, sendo que a **P**-compatibilidade é garantida para os valores iniciais.
- O estado da componente de cada nó só poderá assumir valores dentro do seu território. O protocolo proposto deverá manter a **P**-compatibilidade entre esses territórios, seguindo a idéia de distribuição de um predicado exibida na seção 2.
- Cada nó poderá executar, a qualquer momento, uma restrição voluntária de seu território. Uma vez que isso constitui o que convencionamos chamar de transição segura, poderá ser realizada sem troca de mensagens.
- Ao receber uma mensagem, um nó irá expandir seu território de forma a ser **P**-compatível com aquele representado nessa mensagem, mas considerando também a possível expansão ocorrida no outro nó desde então e seguindo as restrições impostas pelos Compromissos de Progresso.
- O formato das mensagens poderá ser simplificado se pudermos garantir a ***P*-*compatibilidade na diagonal***, ou seja, se for mantida a **P**-compatibilidade entre o território representado na mensagem que chega e o território atual do nó.

O trabalho apresentado em [DI94] descreve um modelo formal de comunicação entre dois nós, adequado para desenvolver provas através de indução finita. Usando esse modelo, é exibida uma demonstração do correto funcionamento do protocolo exibido acima. Em particular, mostra-se que esse protocolo mantém o invariante **P** especificado e ainda possibilita a manutenção da ***P*-*compatibilidade na diagonal***.

5.1 Codificação dos domínios

Com a ***P*-*compatibilidade na diagonal*** garantida, pode-se atribuir uma codificação segura aos elementos dos domínios de subconjuntos **P**-fechados obtidos da seguinte forma: basta que cada subconjunto de um **mesmo** conjunto receba um código diferente. Assim, é possível a cada nó identificar de maneira não ambígua o valor representado pelas mensagens recebidas.

Os valores associados aos subconjuntos das figuras 6 e 7 satisfazem essa condição. Essa identificação das mensagens é designada em [DI94] por *Codificação Unicamente Decifrável*.

Observe como os elementos não maximais da figura 6 recebem códigos que alternam 0 e 1, da mesma forma que a utilizada pelo nó *receptor* do Protocolo AB. Os valores atribuídos aos elementos maximais não são relevantes ao protocolo, como discutimos anteriormente. Um aspecto interessante é que as mensagens do nó *emissor* são codificadas de maneira ligeiramente mais econômica que a proposta no Protocolo AB, e isso foi um resultado conseguido apenas com a aplicação das técnicas propostas.

```
TYPE Σ = {a, b, c};
  E1, E2 = Σ*;
[ P1 (Emissor) :: c1: SET OF E1; e1: E1;
  M1: boolean; U1 : (0,1,2,3); V1, m2 : (0,1);
  c1 := {a,b,c}; e1 := Λ; M1 := false; U1 := 0; V1 := 0;
  *[ M1 = true → SEND(U1);
    | true → c1 := {e1}; M1 := true;
    | e1.a in c1 → e1 := e1.a; V1 := (V1+1) mod 2; U1 := (U1+1) mod 4;
    | e1.b in c1 → e1 := e1.b; V1 := (V1+1) mod 2; U1 := (U1+2) mod 4;
    | e1.c in c1 → e1 := e1.c; V1 := (V1+1) mod 2; U1 := (U1+3) mod 4;
    | RECEIVE(m2), m2 = V1 → c1 := {e1,e1.a, e1.b,e1.c}; M1 := false;
  ]
]
||
[ P2 (Receptor) :: c2: SET OF E2; e2: E2;
  M2: boolean; U2 : (0,1); V2, m1 : (0,1,2,3);
  c2 := {Λ}; e2 := Λ; M2 := false; U2 := 0; V2 := 0;
  *[ M2 = true → SEND(U2);
    | true → c2 := {e2}; M2 := true;
    | e2.a in c2 → e2 := e2.a; U2 := (U2+1) mod 2; V2 := (V2+1) mod 4;
    | e2.b in c1 → e2 := e2.b; U2 := (U2+1) mod 2; V2 := (V2+2) mod 4;
    | e2.c in c1 → e2 := e2.c; U2 := (U2+1) mod 2; V2 := (V2+3) mod 4;
    | RECEIVE(m1), m1 = (V2+1) mod 4 → c2 := {e2,e2.a}; M1 := false;
    | RECEIVE(m1), m1 = (V2+2) mod 4 → c2 := {e2,e2.b}; M1 := false;
    | RECEIVE(m1), m1 = (V2+3) mod 4 → c2 := {e2,e2.c}; M1 := false;
  ]
]
```

Figura 8: Algoritmo equivalente ao protocolo AB.

5.2 Derivação Formal do Protocolo AB

A figura 8 mostra um algoritmo distribuído completo para resolver o problema especificado na seção 4.1. As variáveis auxiliares podem ser adicionadas de maneira semi-automática, de modo a gerenciarem a codificação das mensagens enviadas e recebidas por cada nó. A variável U_i controla o código das mensagens enviadas, enquanto V_i controla o código das mensagens recebidas. A navegação nos domínios de subconjuntos **P**-fechados obedece as restrições impostas pelos *Compromissos de Progresso* discutidos na seção 4.3.

6 Conclusões e Trabalhos Futuros

Usando as técnicas apresentadas, exibimos uma derivação formal de estruturas distribuídas e um algoritmo equivalente ao *Protocolo do Alternating Bit*. Um resultado interessante alcançado foi a obtenção de um formato ligeiramente mais econômico para as mensagens do nó *emissor*. Como salientamos desde o início deste trabalho, o nosso objetivo não era exibir mais uma prova para o Protocolo AB ou uma nova versão para o mesmo. As técnicas que mostramos expressam um processo semi-automático para derivar formalmente algoritmos distribuídos corretamente codificados. Dessa forma, o protocolo genérico exibido na seção 5 pode também ser aplicado

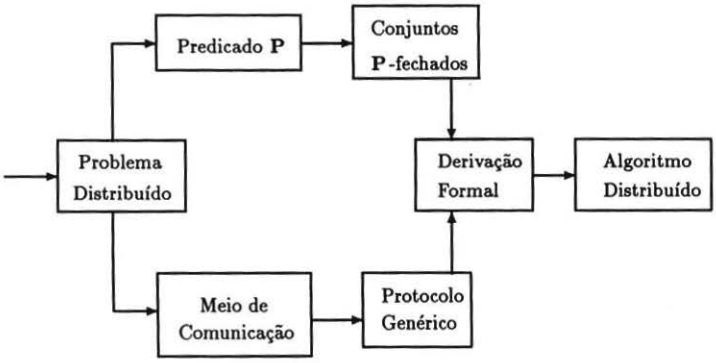

Figura 9: Esquema da derivação formal de algoritmos distribuídos.

em outros problemas envolvendo dois nós que se comunicam em um meio com as características determinadas (ou seja, onde as mensagens podem ser perdidas ou duplicadas, mas a ordem de envio é preservada). Outros exemplos dessa utilização podem ser encontrados em [DI94].

A figura 9 mostra um esquema geral das técnicas propostas. Um problema é especificado através de um predicado **P** envolvendo o estado corrente dos nós. Um protocolo genérico é utilizado para a sincronização, adequado ao meio de comunicação envolvido. Através de **P**, pode-se derivar o formato das estruturas distribuídas, chegando a uma representação dos valores relevantes em um conjunto parcialmente ordenado que designamos *conjunto dos subconjutos* **P**-*fechados*. Aplicando as técnicas desenvolvidas a essas entradas, obtém-se um algoritmo que soluciona o problema proposto. Esse esquema demonstra o nosso principal objetivo: derivar algoritmos distribuídos para problemas especificados por um predicado, cuja correção parcial seja uma conseqüência direta do processo de construção.

Seguindo a linha de pesquisa iniciada em [Car85] e [CDI94], vários outros trabalhos podem ser desenvolvidos. A generalização natural da teoria é a utilização de uma rede com n nós. Esse estudo está sendo conduzido, com a utilização de um grafo acíclico cobrindo toda a rede. Espera-se produzir resultados interessantes, iniciando com a aplicação da técnica ao problema da *Exclusão Mútua*.

A correção parcial do algoritmo descrito na seção 5.2 e de qualquer outro derivado pelo mesmo processo depende diretamente da correção do protocolo genérico definido na seção 5. Como havíamos comentado, o trabalho apresentado em [DI94] descreve um modelo formal de comunicação entre dois nós, adequado para desenvolver provas através de indução finita, com o qual foram demonstradas as propriedades desejadas para esse protocolo. Esse modelo continua sendo objeto de estudos que poderão gerar resultados interessantes, especialmente a forma elegante que propõe para formalizar o meio de comunicação.

Referências

[AS83] Gregory R. Andrews and Fred B. Schneider. Concepts and Notations for Concurrent Programming. *Computing Surveys*, 15(1):3–43, March 1983.

[BST89] Henri E. Bal, Jennifer G. Steiner, and Andrew S. Tanenbaum. Programming Languages for Distributed Computing Systems. *ACM Computing Surveys*, 21(3):261–322,

September 1989.

[BSW69] K.A. Bartlett, R.A. Scantlebury, and P.T. Wilkinson. A note on reliable full-duplex transmission over half-duplex links. *Communications of the ACM*, 12(5):260–261, May 1969.

[Car85] Osvaldo S. F. Carvalho. Une contribution à la programmation des systèmes distribués. Thèse d'État, Université Pierre et Marie Curie (Paris VI), Juillet 1985.

[CDI94] Osvaldo S. F. Carvalho and Vladimir O. Di Iorio. Derivação Formal de Estruturas Distribuídas. In *Anais do VI Simpósio Brasileiro de Arquitetura de Computadores e Processamento Paralelo*, pages 21–35, Caxambu–MG, Junho 1994. Sociedade Brasileira de Computação.

[CR83] Osvaldo S. F. Carvalho and Gerard Roucairol. On mutual exclusion in computer networks. *Communications of the ACM*, 26(2), February 1983.

[DI94] Vladimir Oliveira Di Iorio. Derivação Formal de Estruturas Distribuídas. Dissertação de Mestrado, DCC-UFMG, Abril 1994.

[Dij75] E. W. Dijkstra. Guarded commands, nondeterminancy, and formal derivation of programs. *Communications of the ACM*, 8(18):453–457, August 1975.

[Gri88] Ralph P. Grimaldi. *Discrete and Combinatorial Mathematics.* Addison-Wesley, 1988.

[Hoa69] C. A. R. Hoare. An axiomatic basis for computer programming. *Communications of the ACM*, 10(12):576–580, October 1969.

[Hoa78] C. A. R. Hoare. Communicating sequential processes. *Communications of the ACM*, 21(8), August 1978.

[Kel76] R. M. Keller. "Formal Verification of Parallel Programs". *Communications of the ACM*, 19(7), July 1976.

[Lam78] Leslie Lamport. "Time, Clocks, and the Ordering of Events in a Distributed System". *Communications of the ACM*, 21(7), July 1978.

[Mae85] Mamoru Maekawa. "A $\sqrt{n}$ Algorithm for Mutual Exclusion in Decentralized Systems". *ACM Transactions on Computing Systems*, 3(2), May 1985.

[OG76] S. Owicki and D. Gries. Verifying properties of parallel programs: an axiomatic approach. *Communications of the ACM*, 19(5), May 1976.

[Ore44] Oysten Ore. "Galois Connections". *Transactions of the American Mathematical Society*, 55:493–513, 1944.

[SA86] F.B. Schneider and G.R. Andrews. Lecture notes in computer science vol. 224. In *Current Trends in Concurrency*, pages 669–716. Springer-Verlag, New York, 1986.

[Sha93] A. U. Shankar. "An Introduction to Assertional Reasoning for Concurrent Systems". *ACM Computing Surveys*, 25(3):225–262, September 1993.

[SMS82] Richard L. Schwartz and P. Michael Melliar-Smith. "From State Machines to Temporal Logic: Specification Methods for Protocol Standards". *IEEE Transactions on Communications*, 30(12):2486–2496, December 1982.

[Sza63] G. Szasz. *Introduction to Lattice Theory.* Academic Press, 1963.