LÓGICA DE CONHECIMENTO E EVENTOS EM SISTEMAS ASSÍNCRONOS

Carla Amor Divino Moreira Delgado

TESE SUBMETIDA AO CORPO DOCENTE DA COORDENAÇÃO DOS PROGRAMAS DE PÓS-GRADUAÇÃO DE ENGENHARIA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS EM ENGENHARIA DE SISTEMAS E COMPUTAÇÃO.

Aprovada por:

Prof. Mario Roberto Folhadela Benevides, Ph.D.

Prof. Valmir Carneiro Barbosa, Ph.D.

Prof. Marcelo Finger, Ph.D.

RIO DE JANEIRO, RJ - BRASIL

DEZEMBRO DE 2001

DELGADO, CARLA AMOR DIVINO
MOREIRA

Lógica de Conhecimento e Eventos em Sistemas Assíncronos [Rio de Janeiro] 2001.

VIII, 97 p. 29,7 cm (COPPE/UFRJ, M.Sc., Engenharia de Sistemas e Computação, 2001)

Tese - Universidade Federal do Rio de Janeiro, COPPE

1. Lógica de conhecimento e tempo

I. COPPE/UFRJ II. Título (série)

AGRADECIMENTOS

Agradeço ao Prof. Mario Benevides, meu caríssimo orientador de praticamente toda minha vida universitária, por sua brilhante orientação, seu apoio e incentivo inconteste ao meu trabalho.

Agradeço aos meus familiares, em especial a minha mãe, Sheila, por todo seu carinho e infinita credibilidade no meu sucesso.

Agradeço ao meu queridíssimo namorado, Geraldo Zimbrão da Silva, por todas as horas de apoio, acalento e incentivo, além de sua complacência nas horas mais críticas, e ainda de sua ajuda na estruturação e revisão do texto.

Agradeço aos meus amigos, especialmente Michel, Kate e Vera pelo apoio e companheirismo durante o desenvolvimento deste trabalho.

Agradeço à Vania, por toda ajuda e contribuição que sempre ofereceu de tão boa vontade, e que tanto me foi valiosa.

Agradeço ao professor Valmir, por sua preciosa ajuda na área de sistemas distribuídos assíncronos.

Agradeço a todos os colegas do programa, entre professores, alunos e funcionários, que direta ou indiretamente contribuíram para que essa Tese fosse realizada.

Agradeço a Deus, por continuar me abençoando com oportunidades.

Resumo da Tese apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Mestre em Ciências (M.Sc.)

LÓGICA DE CONHECIMENTO E EVENTOS EM SISTEMAS ASSÍNCRONOS

Carla Amor Divino Moreira Delgado

Dezembro/2001

Orientador: Mario Roberto Folhadela Benevides

Programa: Engenharia de Sistemas e Computação

Este trabalho aborda o tratamento de conhecimento e tempo em sistemas distribuídos através de lógicas, com interesse principal em sistemas fortemente assíncronos de memória distribuída. Para isso, várias lógicas foram revistas e duas novas foram apresentadas, uma para sistemas síncronos com modalidades de ação e outra baseada em eventos para sistemas assíncronos sem falhas. Para esta segunda foram definidos operadores temporais baseados em eventos que dão à linguagem poder de expressão temporal mesmo considerando o tempo sob as poucas restrições cabíveis ao modelo assíncrono. Em especial dois problemas distintos envolvendo conhecimento em sistemas distribuídos assíncronos foram modelados com a linguagem nova, mostrando seu poder de expressão e comprovando sua aplicabilidade.

Abstract of Thesis presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Master of Science (M.Sc.)

KNOWLEDGE AND EVENTS LOGICS IN ASYNCHRONOUS SYSTEMS

Carla Amor Divino Moreira Delgado

December/2001

Advisor: Mario Roberto Folhadela Benevides

Department: Systems Engineering and Computer Science

This work focuses on the handling of knowledge and time in distributed asynchronous systems using logics, specifically on fully asynchronous distributed memory systems. First, a review of some existing logical framework is presented. Second, two new formalism were proposed, one for synchronous systems with action modalities and other based on events to model knowledge evolving in asynchronous systems. To the later logic, a set of event based temporal operators were defined, which provide the language good power of temporal expression, even considering time under the few restrictions allowed to the asynchronous model. Two distinct applications involving knowledge in distributed asynchronous systems were modeled using this new language, showing its expressivity and applicability of the presented formalism.

ÍNDICE

Introdução

Investidas no sentido de formalizar as propriedades do conhecimento apareceram em torno de 1960, motivadas pela comunidade de filosofia. Buscava-se situar a definição de "conhecimento" em meio a todas as outras definições do próprio conhecimento humano, e também estabelecer procedimentos e definições formais para compreender os processos de aprendizado e raciocínio, aquisição de conhecimento, formas de armazenar conhecimento e como utilizar o conhecimento para atingir objetivos ou derivar conclusões.

Ao se tentar entender e analisar as propriedades do conhecimento, filósofos primeiramente tendiam a considerar um único agente. A posteriori, o grande interesse acerca de conhecimento voltou-se para interação entre agentes, ou seja: lidar com outras fontes de conhecimento. Alguns exemplos de sistemas com múltiplos agentes envolvendo conhecimento são computadores em uma rede rodando um protocolo específico, parceiros em um jogo qualquer, membros de uma reunião que visa atingir consenso sobre algum ponto específico, multiprocessadores que paralelizam determinada computação, etc.

Os interesses acerca de representar, adquirir e utilizar conhecimento movimentam cada vez mais recursos na comunidade científica. Pesquisas em áreas de aplicação do conhecimento como lingüística, economia, robótica e inteligência artificial induziram o estudo das relações entre conhecimento e tempo e das relações de conhecimento entre agentes, no caso de sistemas com múltiplos agentes. Em particular, o foco principal do trabalho de FAGIN *et al.* (1995) é entender o processo de raciocínio em um grupo de agentes, mais especificamente estudar o conhecimento em sistemas distribuídos onde os agentes envolvidos são capazes de lidar com o próprio conhecimento, o conhecimento presente no meio, e também o conhecimento de outros agentes.

A contribuição decisiva para a formalização da lógica de conhecimento para um grupo de agentes deve-se a Joseph Y. Halpern, Yoram Moses e Ronald Fagin em meados da década de 80, quando foram publicados os trabalhos "Knowledge and

commom knowledge in a distributed environment" e "A formal model for knowledge, action and communication in distributed systems: preliminary report". Em 1995, uma compilação do trabalho da década foi publicada no livro supracitado "Reasoning about knowledge" de Fagin, Halpern, Moses e Vardi.

A lógica de conhecimento estabelecida nos trabalhos de Halpern, Fagin, Moses e Vardi constitui a lógica modal de conhecimento e crença para múltiplos agentes e baseia-se na noção de "mundos possíveis" de Kripke, onde os mundos possíveis correspondem a estados epistêmicos e as noções de conhecimento e crença correspondem a relações na estrutura relacional de Kripke. O conhecimento em um sistema distribuído porém não pode ser representado apenas por estados epistêmicos estáticos. Como a maioria dos sistemas distribuídos envolve interação, o conhecimento evolui conforme os agentes percebem novas informações e interagem entre si. Alguns exemplos de sistemas com essas propriedades são protocolos de sincronização e cooperação, sistemas de criptografia, jogos e obtenção de acordo.

Algumas propostas para poder representar o conhecimento evoluindo em um sistema distribuído através de lógica modal de conhecimento e tempo para múltiplos agentes foram feitas em LEHMANN (1984) e também em FAGIN *et al.* (1995). Contudo, a maioria dos resultados só pode ser aplicada em sistemas síncronos que correspondem a uma idealização dos sistemas distribuídos reais, onde todos os agentes realizam suas ações utilizando uma base de tempo comum.

O conhecimento em sistemas assíncronos foi explorado em PANANGADEN e TAYLOR (1992), porém o foco de interesse restringiu-se aos operadores modais para conhecimento no grupo, deixando em aberto a forma de manipular o conhecimento que cada agente do sistema tem acerca do tempo de evolução do próprio sistema.

Neste trabalho será abordado o tratamento do conhecimento em sistemas distribuídos assíncronos onde a comunicação entre os agentes envolvidos se dá exclusivamente por intermédio de trocas de mensagens, tanto do ponto de vista da manipulação do conhecimento no grupo de agentes quanto do tratamento temporal do conhecimento no sistema.

Motivação

A noção de conhecimento é uma noção com muitos significados possíveis, mas muito do que computação distribuída faz é de alguma forma, coletivamente

manipular o conhecimento do sistema tal que no final da computação o que os processos sabem individualmente está relacionado de alguma forma com o objetivo original da computação.

Raciocinar sobre as tarefas em um sistema distribuído no nível do conhecimento oferece vantagens como permitir abstração de detalhes específicos de implementação do sistema ou da própria natureza dos agentes. E uma abordagem formal apenas no nível do conhecimento envolvido permite analisar as propriedades de interesse do problema antes de que ele seja implementado. Os resultados mais conhecidos acerca de conhecimento e tempo para sistemas distribuídos contemplam apenas modelos síncronos.

Contribuições

A principal contribuição deste trabalho é a apresentação de um modelo para representar conhecimento em sistemas distribuídos assíncronos, onde os agentes se comunicam através de trocas de menagens. O modelo apresentado permite representar as relações entre conhecimento em um grupo de agentes bem como as relações entre o conhecimento do grupo e o tempo, de acordo com o modelo para sistemas assíncronos de memória distribuída baseado em eventos proposto em LAMPORT (1978).

Estrutura

Apresentamos no capítulo 2 uma revisão do trabalho de FAGIN *et al.* (1995) sobre formalização do conhecimento em um grupo de agentes. Para poder representar o conhecimento evoluindo em um sistema distribuído, apresentamos também no capítulo 2 a lógica modal de conhecimento e tempo para sistemas distribuídos com múltiplos agentes proposta por Lehmann em seu já citado trabalho, fornecendo uma caracterização do papel do tempo na evolução do conhecimento em sistemas síncronos.

No capítulo 3, introduziremos um modelo formal para sistemas distribuídos baseado em eventos. Este modelo servirá como base para a estrutura relacional que suportará a lógica modal de conhecimento para sistemas distribuídos que buscamos. Através dele, podemos descrever as computações distribuídas em termos de propriedades globais, principalmente no que se refere a propriedades temporais no caso assíncrono.

No capítulo 4 propomos uma lógica modal para modelar conhecimento e tempo em sistemas síncronos que utilizando os conceitos de estados e ações condizentes com o modelo para sistemas de memória distribuída apresentado no capítulo 3, que chamaremos de Lógica Dinâmica de Conhecimento. Tal lógica foi objeto de um artigo apresentado no XXI Congresso da Sociedade Brasileira de Computação - Encontro Nacional de Inteligência Artificial: DELGADO *et al.* (2001). Como exemplo, aplicamos LDC para modelar o problema das crianças com lama na testa, um exemplo clássico de sistema baseado em conhecimento que será detalhadamente apresentado.

O capítulo 5 apresenta uma lógica modal de conhecimento e tempo para sistemas distribuídos assíncronos, com o objetivo de representar o conhecimento de um grupo de agentes em um sistema distribuído assíncrono evoluindo no tempo. O modelo baseado em eventos é capaz de lidar com conhecimento e com as possibilidades temporais cabíveis a um algoritmo assíncrono, que dão a noção de tempo particular de cada agente e também do sistema como um todo. Ao final, apresentamos uma aplicação da lógica para modelar alguns exemplos de sistemas distribuídos.

Capítulo 2

Conhecimento em Sistemas Distribuídos

Ao falar de conhecimento em sistemas distribuídos estamos necessariamente falando de conhecimento individual dos agentes do grupo, interações entre os agentes, e evolução do conhecimento. Qualquer sistema interessante envolvendo conhecimento envolverá relações entre conhecimento e tempo e relações de conhecimento entre agentes. Para poder representar o conhecimento em um sistema distribuído, é necessário entender o processo de raciocínio em um grupo de agentes, mais especificamente de agentes capazes de fazer considerações sobre o meio e sobre o raciocínio dos outros agentes.

Um exemplo subjetivo porém simples sobre um agente considerando o conhecimento de outro agente é a venda de um automóvel. O vendedor conhece o valor aproximado do carro, mas isso não é o único fator relevante para ele estipular seu preço. Também é levado em consideração o que ele pensa sobre o que o cliente sabe sobre o valor do carro, afinal é de seu interesse vender o carro pelo mais alto preço que ele consiga fazer o cliente pagar. O cliente, por sua vez, também tem algum conhecimento do valor do carro, e seu interesse é comprar pelo preço mais baixo, sabendo que o vendedor deve querer convencê-lo de que o carro vale mais do que realmente vale. Nesse caso, o preço de venda será estabelecido levando em consideração o conhecimento que os dois agentes têm sobre o sistema, e não será necessariamente o valor exato do carro, que pode inclusive não ser sabido por nenhum dos dois agentes.

É este tipo de situação que buscamos capturar. A interação entre os agentes no decorrer do sistema faz com que novas informações sejam percebidas, novas suposições sejam consideradas e ações sejam tomadas de acordo com o conhecimento cada vez maior envolvido no sistema. Logo, para representar um sistema deste tipo precisamos representar o conhecimento individual de cada agente, e também a configuração destes conhecimentos individuais no decorrer do sistema.

Este capítulo compreende basicamente uma revisão dos conceitos estabelecidos no trabalho de FAGIN *et al.* (1995) acerca de conhecimento de um grupo de agentes em um sistema distribuído, e também uma revisão do trabalho de LEHMANN (1984) sobre conhecimento e tempo em sistemas assíncronos.

2.1 Representação do conhecimento de um grupo de agentes

Utilizaremos a noção de "mundos possíveis" para representar o conhecimento em um grupo de agentes. O conceito intuitivo de mundos possíveis é o seguinte: considere um agente que não tem conhecimento total do sistema. Então, além do estado real do sistema, ele considera também que o sistema possa assumir algum outro estado. Isto é, considerando a informação que ele possui do sistema, ele não é capaz de dizer qual dentre os estados que ele imagina possíveis é o estado real do sistema. Estes possíveis estados do sistema são o que se costuma chamar de "mundos possíveis". O conhecimento de um agente é portanto expresso como relações entre os mundos possíveis. Para clarificar o conceito de mundos possíveis usaremos o seguinte exemplo:

Exemplo: Dois agentes considerando o tempo no Rio e em São Paulo

Imagine uma pessoa (que chamaremos de Agente 1) andando pelas ruas do Rio de Janeiro em um dia de sol. O Agente 1 sabe exatamente o tempo no Rio, mas não sabe dizer se está chovendo ou fazendo sol em São Paulo. Considerando o sistema composto pelo Agente 1 e pelo conhecimento do tempo em São Paulo e no Rio, poderíamos dizer que o Agente 1 considera dois mundos possíveis: o mundo onde faz sol no Rio e chove em São Paulo, e o mundo onde faz sol no Rio e também em São Paulo. Considere agora outra pessoa (Agente 2) que está em São Paulo. O Agente 2 sabe que está chovendo em São Paulo, mas não sabe se no Rio de Janeiro está fazendo sol ou chovendo.

A representação gráfica usual para os mundos possíveis e as relações de conhecimento dos agentes é um grafo, onde os nós correspondem aos estados e as arestas correspondem à relação de conhecimento para o agente indicado no rótulo da aresta. O sistema constituído pelos agentes 1 e 2 e seus conhecimentos e crenças sobre o tempo no Rio de Janeiro e em São Paulo está representado na Figura 2.1.

Os três mundos possíveis: s, t e u, são identificados por círculos. Cada mundo representa um estado de conhecimento, ou seja, um estado epistêmico. Nele

estão descritas as afirmações que valem no mundo em questão. No caso do mundo s, são verdadeiras as afirmações "Faz sol no Rio" e "Chove em São Paulo". As setas indicam as crenças dos agentes sobre os mundos. Dado o estado de conhecimento do Agente 1, ele considera possíveis os mundos s e t, conforme descrito. Mais precisamente, do ponto de vista do Agente 1 os mundos s e t são indistinguíveis, visto que ele não sabe dizer qual dos dois é o mundo real.

Interpretando o grafo, temos que no estado s chove em São Paulo mas o Agente 1 não sabe disso, pois considera possível também o estado t, onde vale que não chove em São Paulo. Já o Agente 2 sabe que chove em São Paulo, pois a proposição "chove em São Paulo" vale em todos os mundos que o Agente 2 considera possível.

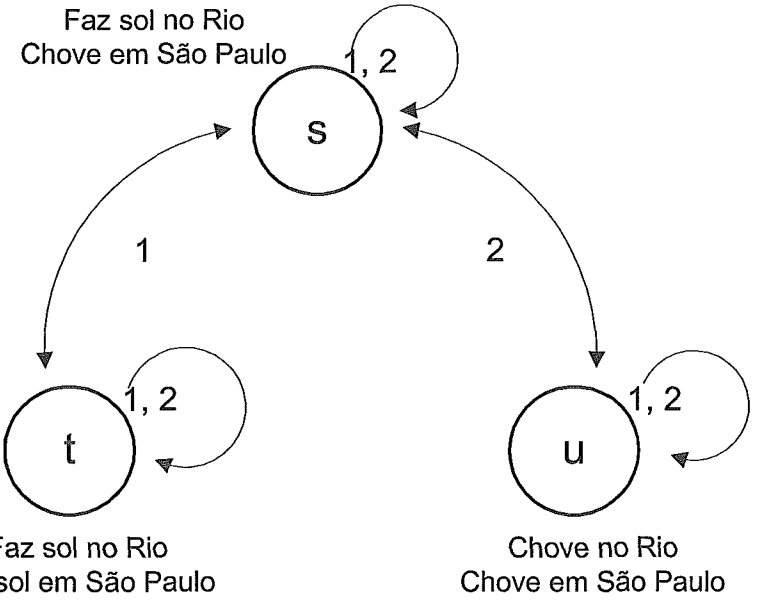

Figura 2.1 – Mundos possíveis para os Agentes 1 e 2 considerando o tempo em São Paulo e no Rio de Janeiro.

No grafo da figura 2.1 temos um ciclo de cada mundo para si próprio, representando a reflexividade da relação de crença: uma vez que um agente está no mundo i, ele considera i possível. Note que a relação de crença em um mundo é reflexiva para todos os agentes: no nosso exemplo, o Agente 2 não considera possível o mundo t, porém ele também aparece na seta da reflexividade de t. As setas são bidirecionais para representar simetria. O caráter simétrico da relação de crença torna-se intuitivo ao se pensar que os estados relacionados são "indistinguíveis" do ponto de vista do agente considerado. Logo, se o Agente 1 não distingue t de s, ele também não distinguirá s de t.

Intuitivamente, quanto menos mundos um agente considerar possível, menor será sua incerteza, e mais ele saberá. Vamos supor que o Agente 1 ouça no rádio que está chovendo em São Paulo; ele não mais irá considerar possível qualquer mundo em que esteja sol em São Paulo. Podemos então retirar o estado s e suas arestas do grafo, conforme mostrado na Figura 2.2.

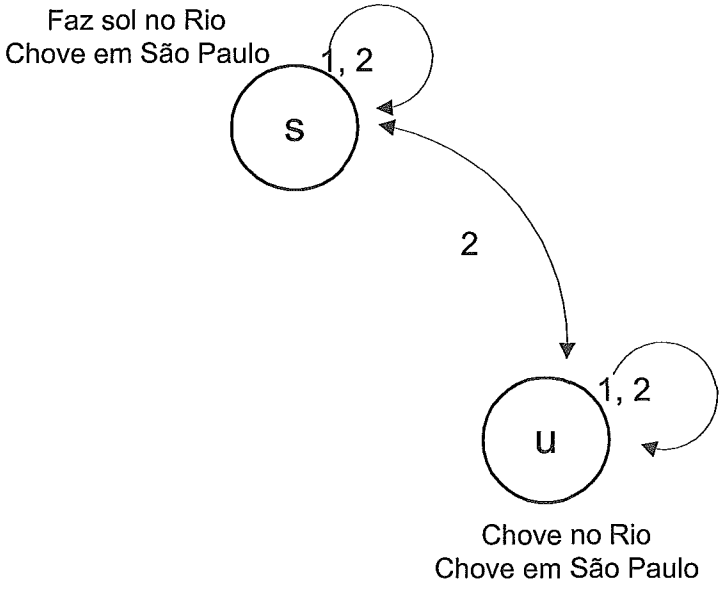

Figura 2.2 – Mundos possíveis para os Agentes 1 e 2 após a aquisição do conhecimento do tempo em São Paulo por parte do Agente 1.

Além de considerar o conhecimento de cada agente do grupo individualmente, em um sistema distribuído estão presentes também relações entre o conhecimento de um agente com os demais agentes envolvidos no grupo. Duas noções importantes em sistemas com múltiplos agentes são as noções de conhecimento comum e conhecimento distribuído.

Conhecimento Comum

Para introduzir a noção de conhecimento comum vamos considerar um sistema do nosso cotidiano que se encaixa perfeitamente neste conceito: o sistema de trânsito. Nesse sistema, todos os motoristas devem saber que o sinal vermelho significa pare. Suponha que possamos assumir que todos os motoristas sabem disso. Um motorista poderia sentir-se seguro? A resposta é não, pois ele poderia pensar que algum outro motorista talvez não soubesse dessa regra e pudesse avançar um sinal vermelho. Para que o sistema funcione, é necessário que todos os motoristas saibam que todos sabem que o sinal vermelho significa pare.

Mesmo a suposição de que "todos sabem que todos sabem algo" pode não ser suficiente para descrever todos os tipos de sistemas com múltiplos agentes envolvendo conhecimento. Há ainda outros tipos de sistemas onde será necessário considerar um estado onde simultaneamente todos sabem sobre um fato f, todos sabem que todos sabem f, todos sabem que todos sabem que todos sabem f, e assim sucessivamente. Essa noção foi primeiramente estudada pelo filósofo David Lewis, no contexto das convenções: ele atentou para o fato de que para que algo seja uma convenção é necessário que esse algo seja de *conhecimento comum* dos membros de um grupo. Considerando a definição intuitiva conhecimento comum seria o que "qualquer um" sabe.

De acordo com HALPERN e MOSES (1990), é a publicação de um fato que o torna de conhecimento comum no grupo. De maneira geral, são possíveis duas formas de se publicar um fato:

1. O fato faz parte das convenções de uma comunidade: no contexto de sistemas distribuídos, as convenções entre os agentes correspondem às informações iniciais comuns do grupo, inseridas antes do início da computação do sistema.
2. O fato é anunciado de forma que todos os agentes estão presentes ao mesmo tempo, e sabem que todos estão presentes: em sistemas distribuídos isto significa fazer com que todos tomem conhecimento do fato simultaneamente.

A noção de conhecimento comum apresentada por HALPERN e MOSES no trabalho supracitado pressupõe simultaneidade, e conseqüentemente só pode ser atingida em sistemas síncronos. Outras noções similares acerca de conhecimento em grupos de agentes foram apresentadas em PANANGADEN e TAYLOR (1992), aplicáveis a sistemas assíncronos.

Conhecimento Distribuído

Para falar sobre conhecimento distribuído vamos considerar outro exemplo: imagine o trabalho de um detetive investigando um crime. Ele busca informações com todas as pessoas envolvidas, busca pistas analisando lugares e objetos e vai chegando a conclusões juntando as informações como um quebra-cabeça. O conhecimento que ele pretende atingir está diluído no sistema, e ele quer analisar o máximo possível do sistema para poder obter todas as informações necessárias. Um grupo tem conhecimento

distribuído de um fato f se o conhecimento está distribuído entre seus membros, isto é, alguém juntando as informações espalhadas entre os agentes do sistema saberá f, mesmo que nenhum agente individualmente saiba f.

O exemplo seguinte fornece uma boa demonstração dos conceitos de conhecimento comum e conhecimento distribuído.

Exemplo: O jogo das três cartas

Em um jogo de cartas com dois ou mais jogadores, ao se distribuir as cartas os jogadores consideram várias possibilidades para as cartas que estão nas mãos dos outros jogadores. No decorrer do jogo, os jogadores vão adquirindo novas informações e chegando a um número bem menor de possibilidades.

Imagine um jogo entre dois agentes 1 e 2 que consista no sorteio de uma carta para cada agente, sobre um total de três cartas no jogo: A, B e C. Cada agente do sistema receberá uma carta, e a terceira carta fica virada para baixo na mesa. Os agentes consideram as configurações possíveis de cartas distribuídas no sistema, ou seja, os estados ou mundos possíveis do sistema de acordo com a carta que tem nas mãos, pois este é o único conhecimento disponível.

Podemos identificar os estados de conhecimento desse sistema como um par ordenado (x_1, x_2), onde x_1 representa a carta que está nas mãos do agente 1 e x_2 representa a carta que está nas mãos do agente 2. O grafo com os mundos possíveis e as relações de crença dos agentes está representado na Figura 2.3 (os ciclos de cada mundo indicando a reflexividade das relações de crença foram omitidos por simplificação).

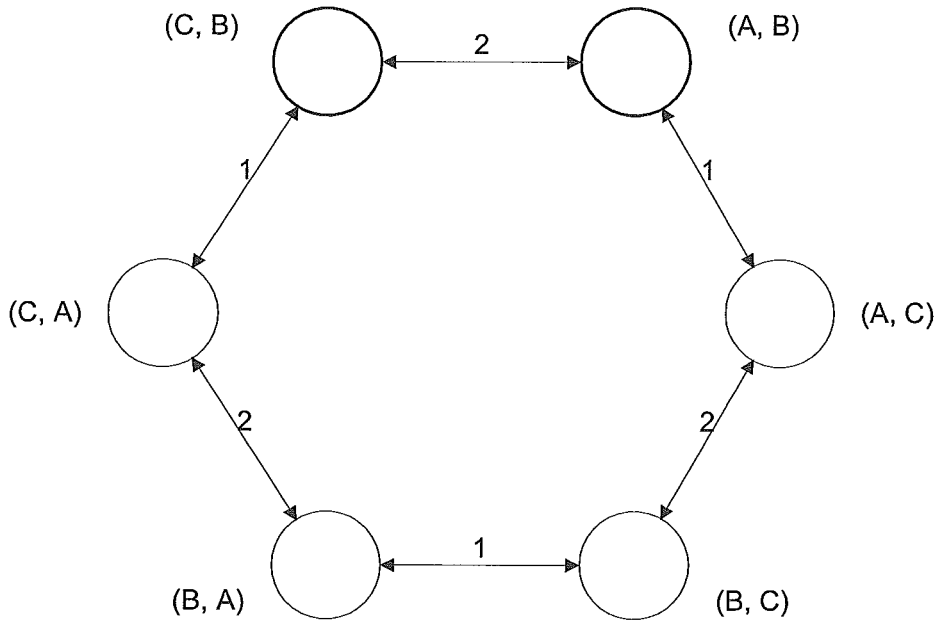

Figura 2.3 – Mundos possíveis para os agentes 1 e 2 no jogo das três cartas A, B e C.

Considere o raciocínio do agente 1 quando ele tem a carta A nas mãos. Nesse caso, ele considera que o agente 2 pode ter a carta B ou a carta C. Considera também que se o agente 2 tem a carta B então o agente 2 pode pensar que o agente 1 tem ou a carta A ou a carta C. E ainda, o agente 1 considera que se o agente 2 tem a carta B, e que se o agente 2 pensa que o agente 1 tem a carta C, então o agente 2 considera possível que o agente 1 pense que o agente 2 tem a carta A. Toda essa informação pode ser obtida percorrendo-se um caminho no grafo. Se existe um caminho conexo entre dois estados do grafo, então seguindo a linha de raciocínio dos agentes segundo o rótulo das arestas (agentes que crêem - ou consideram possíveis - os mundos ligados pelas arestas, que por sua vez admitem ligações a outros mundos sob o ponto de vista de outro agente) podemos chegar a conclusões desse tipo.

Como cada agente tem conhecimento sobre sua carta e sabe as condições da distribuição das cartas no jogo (sabe que há três cartas A, B e C, uma está na mesa e a outra com o outro agente) pode-se dizer que o agente 1 e o agente 2 têm *conhecimento distribuído* da distribuição das cartas.

Imagine agora que a carta da mesa é desvirada e posta na mesa de forma que possa ser vista por todos os jogadores. Como a carta da mesa é revelada simultaneamente a todos os agentes do sistema, ela se torna de *conhecimento comum*. O agente 1 e o agente 2 tomam conhecimento do valor da carta da mesa, e conseqüentemente suas respectivas incertezas acerca da distribuição das cartas são

eliminadas. Sabendo que as cartas envolvidas no sistema e as condições da distribuição também são de conhecimento comum, podemos concluir que o conhecimento comum da distribuição das cartas pode ser obtido no sistema.

2.2 Lógica modal de Conhecimento e Crença

Quando pensamos na forma como o conhecimento é utilizado no nosso mundo constatamos que há poucas verdades absolutas e também poucos sistemas em que os fatos relevantes e seu conhecimento por parte dos agentes envolvidos podem ser completamente descritos. Ao considerarmos a representação de domínios de informação temos que grande parte das vezes a ausência de informação faz com que o que haja de mais relevante disponível no sistema seja a consideração de "possibilidades" ou "crenças", conforme visto nos exemplos 2.2.1 e 2.2.2 anteriores.

As noções de possibilidade ou crença são o objeto principal do estudo da lógica modal. Segundo HUGHES e CRESSWELL (1996) a lógica modal pode ser descrita resumidamente como a lógica da necessidade e da possibilidade, do que "deve ser" e do que "pode ser"; em contraste com o mapeamento direto de proposições em verdadeiro e falso que a lógica proposicional oferece.

Para o tratamento de conhecimento em sistemas com múltiplos agentes, a lógica modal adequada seria a lógica de conhecimento e crença, do que um agente "sabe" e do que um agente "acredita" (considera possível). Como se tornou claro nos exemplos supracitados, as noções de conhecimento e crença estão fundamentadas nas relações de indistinguibilidade que interligam os mundos possíveis de Kripke. O uso de interpretações em estruturas relacionais (semântica relacional ou de Kripke) para explicar a estrutura lógica dos sistemas modais é um dos temas fundamentais no estudo matemático da lógica modal. Segundo BLACKBURN *et al.* (1999) as linguagens modais são uma forma simples e expressiva de falar sobre estruturas relacionais.

Linguagens modais concedem uma perspectiva local interna em estruturas relacionais, visto que as fórmulas modais são avaliadas dentro das estruturas, em um estado em particular. A função dos operadores modais é permitir que a informação armazenada nos outros estados seja rastreada, mas apenas os estados acessíveis do ponto em questão por uma transição apropriada podem ser acessados dessa forma. Para estruturas relacionais de representação do conhecimento em grupos de agentes essa perspectiva corresponde ao ponto de vista local do agente levando em conta o estado de

conhecimento em que ele se encontra, o que é uma forma bem realista de modelar a noção de conhecimento dos agentes de um sistema distribuído.

Apresentaremos a seguir uma linguagem, seu respectivo modelo semântico e um sistema axiomático correspondente para representar uma lógica de conhecimento e crença para múltiplos agentes de um sistema distribuído.

2.2.1 Sintaxe para lógica de conhecimento e crença

A linguagem utilizada é a da lógica proposicional modal para um número m de agentes. Chamaremos essa linguagem de L_m.

Símbolos

Os símbolos da lógica modal proposicional L_m são:

1. Um conjunto Φ enumerável de símbolos proposicionais;
2. Pontuação: "(" e ")";
3. Conectivos: "$\neg$", "$\wedge$", "$\vee$" e "$\rightarrow$" ;
4. Modais: K_i e B_i, i=1, 2, 3, ..., m (um para cada agente);

Os operadores de L_m são definidos da seguinte forma:

1. Os conectivos seguem as definições da lógica proposicional;
2. O operador modal "K_i" indica conhecimento em relação ao agente *i*, ou seja, $K_i\varphi$ indica que o agente i sabe φ, para i=1, 2, 3, ..., m;
3. O operador modal "B_i" indica crença em relação ao agente i, ou seja, $B_i\varphi$ indica que o agente i acredita em φ, para i=1, 2, 3, ..., m.

Fórmulas

As fórmulas da linguagem são descritas pelas seguintes regras:

1. Todo símbolo proposicional de Φ é uma fórmula, chamada fórmula atômica;
2. Se φ é uma fórmula, então $(\neg\varphi)$ também é uma fórmula;
3. Se φ e ψ são fórmulas, então $(\varphi \wedge \psi)$, $(\varphi \vee \psi)$ e $(\varphi \rightarrow \psi)$ também são fórmulas;
4. Se φ é uma fórmula, então $K_i\varphi$ e $B_i\varphi$ também são fórmulas, para i=1, 2, 3, ..., m;
5. Nada é uma fórmula, a não ser que seja forçado por um dos itens acima.

2.2.2 Semântica para lógica de conhecimento e crença

Para a representação de sistemas desse tipo faz-se necessária uma estrutura que contenha as noções de mundos, mundos possíveis e acessíveis e a valoração nesses mundos. Sob esse enfoque pode-se definir lógica modal (bem como a lógica de conhecimento e crença) como um formalismo para lidar com estruturas relacionais.

Uma estrutura como essa foi proposta por KRIPKE (1959, 1963).

L_m-*frame*

Um L_m-*frame* $F = (S,R_i)$, i=1, 2, 3, ..., m é uma estrutura onde:

1. S é um conjunto de estados ou mundos possíveis;
2. R_i é uma relação binária em S, ou seja, um conjunto de pares de elementos de S onde i=1, 2, 3, ..., m ($R_i \subseteq S \times S$).
3. A relação Ri é reflexiva, simétrica e transitiva.

Modelo

Um modelo de Kripke sobre $F = (S, R_i)$ onde i=1, 2, 3, ..., m é um par $M = (F, \pi)$, onde π é uma interpretação (função de valoração) que associa valores verdade às primitivas de Φ em cada estado de S, isto é, $\pi: \Phi \times S \rightarrow \{V, F\}$.

Satisfatibilidade

Uma fórmula φ de L_m é verdadeira em (M, s), ou seja, é verdadeira em um estado $s \in S$ para um modelo M quando:

1. $M, s \models p$ se e somente se $\pi(s, p)=V$, onde $p \in \Phi$;
2. $M, s \models \neg\varphi$ se e somente se não é o caso que $M, s \models \varphi$;
3. $M, s \models \varphi \wedge \psi$ se e somente se $M, s \models \varphi$ e $M, s \models \psi$;
4. $M, s \models K_i\varphi$ se e somente se para todo $t \in S$ tal que $(s, t) \in R_i$ temos que $M, t \models \varphi$;
5. $M, s \models B_i\varphi$ se e somente se existe t tal que $t \in S$ e $(s, t) \in R_i$, e $M, t \models \varphi$.

Seja $M = (F, \pi)$ um modelo para F. Dizemos que M satisfaz φ se existe algum mundo $s \in S$ tal que $M, s \models \varphi$. Dizemos que φ é satisfatível se existe algum modelo que o satisfaça, caso contrário, dizemos que φ é insatisfatível. Uma fórmula φ é válida em um Modelo M se φ é satisfeita em todos os estados de M (para todo $s \in S$, M,

s |= φ). Uma fórmula φ é válida em um *frame* F se φ é válida em todos os modelos sobre F (para todos M e s, M, s |= φ).

2.2.3 Sistemas axiomáticos do conhecimento

Axioma Dual

O operador modal "B_i" pode ser definido em função do operador modal "K_i", para i=1, 2, 3, ..., m , dado o axioma dual:

- $B_i\varphi \leftrightarrow \neg K_i \neg \varphi$, para i=1, 2, 3, ..., m.

Sistema K_m

O sistema K_m consiste de:

- Dois axiomas:
  1. Todas as tautologias do cálculo proposicional ;
  2. $K_i(\varphi \rightarrow \psi) \rightarrow (K_i\varphi \rightarrow K_i\psi)$, para todo φ e $\psi \in L_m$ e i=1, 2, ..., m.
- Duas regras de inferência:
  1. De φ e $\varphi \rightarrow \psi$ derive ψ (*modus ponens*);
  2. De |= φ derive $K_i\varphi$ (generalização do conhecimento).

Teorema: O sistema K_m juntamente com o axioma dual é uma axiomatização correta e completa para os mundos de Kripke (DOLEV *et al.*, 1985).

Outros axiomas importantes que caracterizam conhecimento, restringindo a relação R_i:

Axioma A3

Axioma do conhecimento: apenas fatos verdadeiros são conhecidos

$K_i\varphi \rightarrow \varphi$

Axioma A4

Introspecção positiva: o agente tem conhecimento de seu conhecimento;

$K_i\varphi \rightarrow K_i\, K_i\varphi$

Axioma A5

Introspecção negativa: o agente tem conhecimento de sua falta de conhecimento.

$\neg K_i\varphi \rightarrow K_i \neg K_i\varphi$

Teorema: A3, A4 e A5 só são válidos se a relação R_i for respectivamente reflexiva, transitiva e simétrica.

Outros sistemas axiomáticos

- $T_m = K_m + A3$
- $S4_m = T_m + A4$
- $S5_m = S4_m + A5$

Teorema: T_m é uma axiomatização correta e completa para os mundos reflexivos;

Teorema: $S4_m$ é uma axiomatização correta e completa para os mundos reflexivo-transitivos;

Teorema: $S5_m$ é uma axiomatização correta e completa para os mundos reflexivo-simétrico-transitivos (DOLEV *et al.*, 1985);

2.2.4 Estados de conhecimento de um grupo de agentes

Até agora, a linguagem descrita não permite expressar as noções de conhecimento comum e distribuído, além de outras noções interessantes em um sistema com múltiplos agentes. A seguir, formalizamos os estados de conhecimento relativos a um grupo G de agentes, e apresentamos os axiomas que expressam as propriedades desses estados.

Conhecimento de um agente: $A_G\varphi$

Conhecimento relativo a algum agente do grupo sobre um fato φ.

$(M, s) \models A_G\varphi$ se e somente se existe o agente i tal que $i \in G$ e para todo t, se $(s, t) \in R_i$ então $(M, t) \models \varphi$;

Conhecimento distribuído: $D_G\varphi$

Unindo o conhecimento individual de todos os agentes do grupo, podemos deduzir φ.

$(M, s) \models D_G\varphi$ se e somente se para todo t, se $(s, t) \in \cap_{i \in G} R_i$ então $(M, t) \models \varphi$;

Conhecimento de todos: $E_G\varphi$

Todos os agentes do grupo têm conhecimento do fato φ.

$(M, s) \models E_G\varphi$ se e somente se para todo i tal que $i \in G$ então $(M, s) \models K_i\varphi$, ou seja:

$(M, s) \models E_G\varphi$ se e somente se para todo i tal que $i \in G$ então para todo t, tal que $(s, t) \in R_i$ então $(M, t) \models \varphi$;

Conhecimento comum: $C_G\varphi$

Um fato φ é de conhecimento comum em um grupo se e somente se φ é verdadeiro e: todo mundo no grupo sabe φ, todo mundo sabe que todo mundo sabe φ, todo mundo sabe que todo mundo sabe que todo mundo sabe φ, etc.

Seja $E_G^0\varphi$ uma representação para φ, e seja $E_G^{k+1}\varphi$ uma representação para $E_GE_G^k\varphi$, para $k \geq 1$. Em particular, $E_G^1\varphi$ é uma representação para $E_G\varphi$.

$(M, s) \models C_G\varphi$ se e somente se $(M, s) \models E_G^k\varphi$, para k=1, 2, 3, ...

Axiomas para conhecimento em grupo de agentes

Considerando a adição dos operadores D_G, E_G e C_G na linguagem, a semântica de Kripke deve incluir as condições descritas acima, decorrentes da definição destes operadores. Como conseqüência, acrescentam-se os seguintes axiomas e regras nos sistemas axiomáticos:

- D1. $K_i\varphi \rightarrow D_G\varphi$
- D2. $D_G\varphi \rightarrow D_{G'}\varphi$ se $G \subseteq G'$
- C1. $E_G\varphi \rightarrow \wedge_{i \in G} K_i\varphi$
- C2. $C_G\varphi \rightarrow E_G(\varphi \wedge C_G\varphi)$
- H1. $C_G\varphi \rightarrow E_G^k\varphi \rightarrow E_G\varphi \rightarrow A_G\varphi \rightarrow D_G\varphi \rightarrow \varphi$ (hierarquia de estados de conhecimento)
- RC1. De $\models \varphi \rightarrow E_G(\psi \wedge \varphi)$ derive $\models \varphi \rightarrow C_G\psi$ (regra da indução).

Teorema: (FAGIN *et al.*, 1995)

Para a linguagem incluindo os operadores D_G, E_G e C_G, o sistema que incorpora seus axiomas e regras a K_m (T_m, $S4_m$ e $S5_m$) é correto e completo para mundos de Kripke (reflexivos, reflexivo-simétricos, reflexivo-simétrico-transitivos).

A partir das definições apresentadas anteriormente, podemos pensar os mundos possíveis como sistemas paralelos da lógica clássica, cada um com suas próprias fórmulas e proposições. Os mundos possíveis se relacionam através das crenças dos agentes. Quando um agente em determinado mundo sabe um fato, então nesse

mundo esse agente considera possíveis apenas outros mundos em que esse fato é verdadeiro. De uma forma análoga, se um agente acredita em um fato quando está em determinado mundo, esse agente considera que existe pelo menos um outro mundo possível em que essa sentença é verdadeira.

Assim, a lógica de conhecimento e crença permite expressar as relações de conhecimento e crença de um grupo de agentes sobre mundos predefinidos, o que viabiliza a modelagem estática do conhecimento dos agentes de um sistema. Ao se pensar em um sistema distribuído dinâmico, poderíamos utilizar a lógica de conhecimento e crença para representar o conhecimento dos agentes do sistema em um "instante" no tempo, ou seja: seria como obter uma fotografia do estado do conhecimento no sistema distribuído. Para representar as modificações que o conhecimento de um sistema distribuído sofre ao longo da evolução do sistema faz-se necessário um modelo um pouco mais elaborado, como será visto adiante.

2.3 Evolução do conhecimento em sistemas distribuídos

Muitos sistemas distribuídos são caracterizados pela evolução do conhecimento ao longo de sua execução. À medida em que o tempo passa, novas informações são adquiridas pelos agentes e consequentemente suas assertivas acerca do conhecimento envolvido no sistema mudam. Dessa forma, o agente aprimora sua percepção acerca do real estado do sistema e vai gradativamente descartando suas incertezas. Alguns exemplos de sistemas com essas propriedades são protocolos de sincronização e cooperação, sistemas de criptografia, jogos e obtenção de acordo.

Para poder representar o conhecimento evoluindo em um sistema distribuído uma lógica modal de conhecimento e tempo para sistemas distribuídos com múltiplos agentes foi descrita em LEHMANN (1984). O trabalho apresenta um sistema axiomático completo para conhecimento comum e a uma caracterização precisa do papel do tempo na evolução do conhecimento. A linguagem, capaz de expressar os conceitos de conhecimento de um agente e do grupo no decorrer do tempo, será apresentada a seguir. Através dela podemos falar do conhecimento que um agente tem sobre o estado atual do sistema (presente), dos estados futuros e do conhecimento que os outros agentes podem ou não ter sobre os estados presente e futuro do sistema.

As próximas seções apresentam a linguagem, seu modelo semântico e o sistema axiomático envolvendo as relações entre conhecimento e tempo para a lógica de

conhecimento e tempo em um sistema distribuído com múltiplos agentes. O modelo para sistemas distribuídos utilizado pressupõe que o sistema seja síncrono.

2.3.1 Sintaxe para lógica de conhecimento e tempo

A linguagem é a mesma da lógica de conhecimento e crença para sistemas distribuídos acrescida dos seguintes operadores temporais:

Operadores Temporais

1. Sempre no futuro: $\Box$
2. No próximo passo: O
3. Até que: U ($\varphi U \psi$ se φ é verdade até que ψ seja verdade)

Fórmulas

As fórmulas da linguagem são descritas segundo as mesmas regras da lógica de conhecimento e crença, acrescida da seguinte regra:

- Se φ e ψ são fórmulas então $\Box\varphi$, $O\varphi$, $\varphi U \psi$ também são fórmulas.

Permitir estabelecer relações entre conhecimento e tempo dá à linguagem forte poder de expressão. Observemos alguns exemplos: A fórmula $O(K_i\varphi \vee K_i\neg\varphi)$ expressa que no próximo passo o agente i saberá se φ é verdadeiro ou falso. A fórmula $C_G\Box(\varphi \leftrightarrow O\varphi)$ significa que é de conhecimento comum que a validade de φ independe do tempo, ou seja, φ é invariante no tempo.

Podemos definir o conectivo E_G (conhecimento de todos) a partir de K_i da seguinte forma: $E_G\varphi = \wedge_{\forall i=1,\,..\,m} K_i\varphi$. A fórmula $\varphi \rightarrow OE_G\varphi$ significa que se φ vale no presente então também valerá no próximo passo e então todos saberão que φ vale. A fórmula $C_G\Box(\varphi \rightarrow OE_G\varphi)$ implica que a partir do momento que φ se tornar verdade (caso isso aconteça), o conhecimento de φ se difunde no sistema, ou seja, após t pulsos do relógio $E_G^t\varphi$ será verdade, para todo t.

2.3.2 Modelo semântico para conhecimento e tempo em sistemas distribuídos

Podemos encarar a evolução de um sistema distribuído como um *frame* de uma estrutura Kripke. Como conseqüência da incorporação de tempo ao modelo, um estado global do sistema (ou mundo possível) determina também um estado temporal para o sistema. Além disso, o conhecimento de um agente deve ser considerado sob suas

propriedades temporais: intuitivamente, acredita-se que um agente dispõe no estado atual de todo o conhecimento adquirido no passado. Consideramos que o conjunto de todos os estados de conhecimento passados de um agente determina sua *história*. O resultado é que modelo semântico para a lógica de conhecimento e tempo em sistemas distribuídos baseia-se em uma estrutura Kripke, porém com as devidas adequações para incorporar as propriedades temporais.

Modelo de Kripke para conhecimento e tempo

Dado um conjunto Φ de proposições primitivas que descrevem os fatos elementares do sistema (aqui assumimos por simplicidade que as propriedades elementares do sistema podem ser descritas adequadamente com lógica proposicional; a extensão do *framework* para lógica de primeira ordem pode ser feita sem perda de generalidade), um modelo $M=(S, w, \pi, \sim_1, \sim_2, ..., \sim_n)$ é uma estrutura onde:

1. S é um conjunto cujos elementos são interpretados como visões instantâneas do estado do sistema e denominados estados; uma história é uma seqüência infinita de estados, isto é, um membro de S^N;
2. $w \in S^N$ e será chamado de história real;
3. π é uma interpretação (função de valoração) que associa valores verdade às primitivas de Φ em cada estado de S.
4. Para todos os agentes i, $\sim_i$ é uma seqüência infinita de relações de equivalência , ou seja para todo $k \in N$, $\sim_i^k$ é uma relação de equivalência em S^N, o conjunto de todas as histórias possíveis.

O significado intuitivo de $\sim_i^k$ é: duas histórias σ e τ são equivalentes com respeito a i e k se o conhecimento que uma pessoa i adquiriu sobre o mundo até o instante k não lhe permite distinguir entre σ e τ. Em outras palavras, se a história real é σ , então para tudo que a pessoa i sabe até o instante k a história real poderia também ser τ.

Incluímos no sistema a exigência de que os agentes não esquecem o que já aprenderam, o que significa que: se o agente i sabe o suficiente no tempo k para distinguir entre as histórias σ e τ, ele poderá, a qualquer instante n no futuro (n>k), distinguir entre σ e τ. Definimos então que para todo agente i e todo $k \in N$, temos $\sim_i^{k+1} \subseteq \sim_i^k$. A inclusão dessa exigência no sistema impõe que os agentes estejam cada vez mais aptos a distinguir entre histórias diferentes conforme o conhecimento evolui no decorrer do tempo, e as relações de equivalência correspondentes se refinam.

Vamos definir também a relação R_k sobre histórias: $R_k = (\cup_{\forall i} \sim_i^k)^*$, onde * representa o fecho reflexivo e transitivo da relação. Ela será usada na definição de satisfatibilidade para o conectivo C_G.

Satisfatibilidade

Agora as fórmulas da linguagem serão avaliadas em um modelo $M=(S, w, \pi, \sim_1, \sim_2, ..., \sim_n)$. Para uma história σ e um instante $k \in N$.

1. $(M, \sigma, k) \models p$ se e somente se $\pi(\sigma, k)(p)=V$, onde $p \in \Phi$;
2. $(M, \sigma, k) \models \neg\varphi$ se e somente se não é o caso que $(M, \sigma, k) \models \varphi$;
3. $(M, \sigma, k) \models \varphi \wedge \psi$ se e somente se $(M, \sigma, k) \models \varphi$ e $(M, \sigma, k) \models \psi$;
4. $(M, \sigma, k) \models O\varphi$ se e somente se $(M, \sigma, k+1) \models \varphi$;
5. $(M, \sigma, k) \models \Box\varphi$ se e somente se $(M, \sigma, n) \models \varphi$ para todo $n \geq k$;
6. $(M, \sigma, k) \models \varphi U \psi$ se e somente se para todo $n \geq k$ tal que $(M, \sigma, n) \not\models \varphi$ existe algum m tal que $k \leq m \leq n$ tal que $(M, \sigma, m) \models \psi$;
7. $(M, \sigma, k) \models K_i\varphi$ se e somente se para todas as histórias σ tais que $\sigma \sim_i \tau$ temos que $(M, \tau, k) \models \varphi$;
8. $(M, \sigma, k) \models C_G\varphi$ se e somente se para todas as histórias τ tais que $\sigma R_k \tau$ temos que $(M, \tau, k) \models \varphi$;

Lema: Para toda fórmula φ, modelo M, instante k e história σ conforme descrito anteriormente, temos que $(M, \sigma, k) \models C_G\varphi$ se e somente separa qualquer $m \in N$ e para qualquer seqüência $i_1, i_2, ..., i_m$ de agentes temos que $(M, \sigma, k) \models K_{i1}K_{i2}...K_{im}\varphi$.

Axiomas

As relações de conhecimento $\sim_i$ são interpretadas como relações de equivalência, o que corresponde a utilizar $S5_m$ como sistema axiomático base para a linguagem. O seguinte conjunto de axiomas dá a noção de validade acima apresentada:

A1. $K_i(\varphi \rightarrow \psi) \rightarrow K_i\varphi \rightarrow K_i\psi$ para i=1, 2 ... m.

A2. $C_G(\varphi \rightarrow \psi) \rightarrow C_G\varphi \rightarrow C_G\psi$

A3. $K_i\varphi \rightarrow \varphi$ para i=1, 2 ... m.

A4. $K_i\varphi \rightarrow K_i K_i\varphi$ para i=1, 2 ... m.

A5. $C_G\varphi \rightarrow \varphi$

A6. $\neg K_i\varphi \rightarrow K_i\neg K_i\varphi$ para i=1, 2 ... m.

A7. $C_G\varphi \rightarrow K_iC_G\varphi$ para i=1, 2 ... m.

A8. $C_G(\varphi \rightarrow E_G\varphi) \rightarrow \varphi \rightarrow C_G\varphi$

A9. $K_iO\varphi \rightarrow OK_i\varphi$ para i=1, 2 ... m.

A10. $C_GO\varphi \rightarrow OC_G\varphi$

Regras de inferência

R0. Modus Ponens: De φ e $\varphi \rightarrow \psi$ derive ψ

R1. Generalização de $\Box$ (sempre no futuro): De $\models \varphi$ derive $\Box\varphi$

R2. Generalização do conhecimento comum: De $\models \varphi$ derive $C_G\varphi$

Teorema: O conjunto de regras e axiomas acima é uma axiomatização correta e completa para a linguagem de conhecimento e tempo definida acima para sistemas distribuídos síncronos. Prova: LEHMANN *et al.* 1982.

Vejamos agora significados e justificativa para os axiomas e regras de inferência apresentados. O axioma A1 e a regra R2 expressam que estamos lidando com pessoas inteligentes, e que todos sabem que os envolvidos no sistema são inteligentes. O axioma A2 é tecnicamente necessário e justifica-se por definição do operador C_G. A3 é o axioma do conhecimento apresentado anteriormente, e A4, tecnicamente necessário, advém de A3. A5 é o axioma para introspecção negativa já visto, e A6 é conseqüência da definição de C_G. A7 é o axioma de ponto fixo de C_G, A8 representa que os agentes não esquecem os fatos sabidos, e A9 representa o similar de A8 para conhecimento comum.

O sistema acima apresenta uma axiomatização completa para as propriedades características de conhecimento comum, e fornece meios de lidar com tempo, dando à linguagem poder para expressar a evolução do conhecimento no tempo. Porém pressupõe que o tempo considerado no sistema seja discreto e que o sistema seja síncrono, ou seja: que haja um relógio global capaz de informar simultaneamente a todos os agentes sobre o término de um pulso e o início de outro. Se assim não fosse, os operadores de conhecimento comum - C_G e próximo passo global - O não fariam sentido, devido à forma como foram definidos (suas definições exigem simultaneidade).

Apesar disso, a lógica de conhecimento e tempo proposta por Lehmann nos permite analisar certas propriedades acerca do estado de conhecimento do grupo ao longo da execução do sistema sem impor restrições a forma como os agentes adquirem o conhecimento.

Em LEHMANN (1984) pode ser visto um exemplo de utilização da lógica apresentada aplicada a uma das várias versões do problema das crianças com lama na testa. O exemplo, de natureza síncrona, fornece uma análise precisa do papel do tempo na evolução do conhecimento em um sistema distribuído, e evidencia as propriedades temporais que podem ser expressas através da linguagem.

Capítulo 3

Sistemas Distribuídos

No capítulo anterior vimos que grande parte dos sistemas distribuídos envolvendo conhecimento engloba modificações dos estados de conhecimento dos agentes envolvidos, conforme se dá a evolução do sistema. Em um modelo natural de sistema distribuído envolvendo conhecimento, os agentes vão incrementando seu conhecimento sobre o real estado do sistema a medida que eventos ocorrem e a comunicação flui na computação.

O foco deste capítulo é apresentar uma metodologia para especificação formal de sistemas distribuídos e para o desenvolvimento de algoritmos para esta classe de sistemas. Essa especificação é de especial importância para o trabalho desenvolvido, pois constitui a base para o tratamento que será dado à evolução do conhecimento em sistemas distribuídos nos capítulos posteriores.

Dentre as configurações de sistemas distribuídos, veremos a seguir os sistemas de memória distribuída, que possuem a característica de que os agentes só se comunicam através de trocas de mensagens. Essa é uma das características desejáveis em nosso modelo, pois consideramos que os agentes envolvidos em uma computação distribuída são completamente independentes do ponto de vista físico, ou seja, não compartilham nenhum recurso além dos canais de comunicação da rede. Nas seções seguintes será apresentado um modelo teórico de arquitetura e descrição dos sistemas de memória distribuída. A motivação para essa abordagem fundamentalmente teórica é que ela nos permite abstrair das restrições de implementação dos sistemas do mundo real, dando relevância apenas às características comuns a toda classe de sistemas do nosso interesse. O modelo apresentado a seguir originalmente proposto em LAMPORT (1978) foi revisto em BARBOSA (1996).

3.1 Modelo de arquitetura para sistemas de memória distribuída

Sistema de memória distribuída

Segundo BARBOSA (1996), um sistema de memória distribuída supõe um conjunto de processadores interconectados de alguma forma por uma rede de canais de comunicação (a topologia da rede de comunicação é definida de acordo com o sistema sendo considerado). Por definição os processadores não compartilham nenhuma memória fisicamente, conseqüentemente toda a comunicação entre eles deve necessariamente acontecer por intermédio de trocas de mensagens através dos canais de comunicação da rede. Usaremos neste capítulo os termos sistemas de memória distribuída e sistemas de troca de mensagens indistintamente.

Representaremos um sistema de memória distribuída por um grafo conexo direcionado $\mathbf{G} = (\mathbf{N}, \mathbf{E})$, onde o conjunto de nós $\mathbf{N}$ do grafo representa o conjunto de processadores e o conjunto de arestas $\mathbf{E}$ é o conjunto de canais unidirecionais interligando os processadores, conforme ilustrado na figura 3.1. Dois processadores se comunicam diretamente somente se existe um canal de comunicação entre eles, no sentido origem-destino da comunicação. Caso contrário, a comunicação se dá em função do repasse de mensagens através de um caminho entre os processadores de origem e destino sobre os canais da rede.

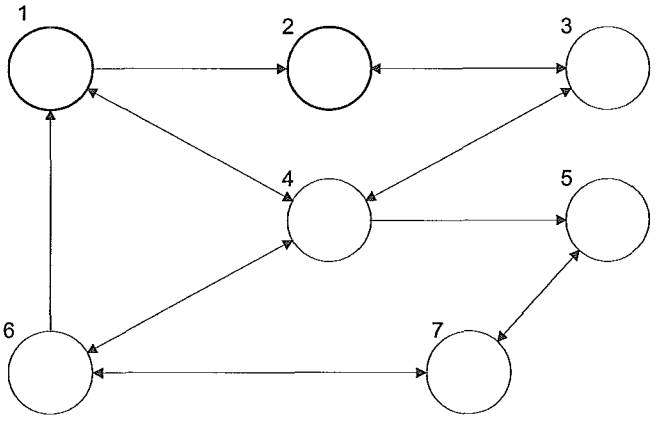

Figura 3.1: Grafo G=(N, E) representando o sistema de memória distribuída composto pelos processadores (ou nós) N={1, 2, 3, 4, 5, 6} e os canais de comunicação E={(1, 2), (6, 1), (1, 4), (4, 1), (2, 3), (3, 2), (3, 4), (4, 3), (4, 5), (4, 6), (6,4), (5, 7), (7,5), (6, 7), (7,6)}

Programa para sistemas de memória distribuída

Consideramos um programa para um sistema de memória distribuída como composto por uma coleção de entidades de código seqüencial cada uma sendo executada em um processador, talvez mais de uma no mesmo processador. Tais programas são usualmente denominados tarefas, processos ou *threads* (usaremos os termos tarefa e programa para referenciá-los). Cada processador executa individualmente seus programas, e a execução em paralelo de programas afins nos processadores dita o comportamento do sistema distribuído.

Comunicação em sistemas de memória distribuída

Conforme dito anteriormente, as tarefas se comunicam através de trocas de mensagens sobre os canais da rede de comunicação. Isso significa que a comunicação entre duas tarefas pode se dar de três formas distintas. Quando os processadores de orígem e destino da mensagem são o mesmo todo o processo de troca de mensagem é viabilizado pelo próprio processador, o que significa que as mensagens não precisam passar por nenhum canal de comunicação ao percorrer o caminho entre orígem e destino. Quando existe uma aresta de **E** direcionada do processador que hospeda a tarefa de orígem para o de destino da mensagem, a troca de mensagens será viabilizada apenas pelos processadores envolvidos na comunicação. Porém para realizar a troca mensagens entre duas tarefas que são executadas em processadores distintos não conectados diretamente por canais de comunicação é necessário delegar parte do processo de comunicação a outros processadores do sistema, de forma que estes viabilizem o

repasse de mensagens sob um caminho entre os processadores de orígem e destino da mensagem.

Em vista disso, os processadores do sistema além de executar as suas tarefas devem prover o tráfego de mensagens na rede de comunicação, mesmo que nenhuma de suas tarefas esteja envolvida na comunicação em questão. Para isso, cada processador precisa ser dotado de funcionalidades específicas para lidar com o tráfego de mensagens na rede que passa através de seus canais de comunicação adjacentes sem que isso venha a interferir no processamento local. Para refletir a independência entre essas duas funções do processador modelaremos em nossa arquitetura o **processador** ou **agente** de um sistema de memória distribuída como uma estrutura na verdade composta por dois processadores:

- O **processador principal** (ou *host*), que é responsável pela execução das tarefas ou processos que são executados no agente;
- O **processador de comunicação**, um outro processador independente do processador principal que se responsabiliza pelo gerenciamento das funções de comunicação;

A arquitetura do sistema de memória distribuída da figura 3.1 aparece na figura 3.2, conforme a especificação apresentada.

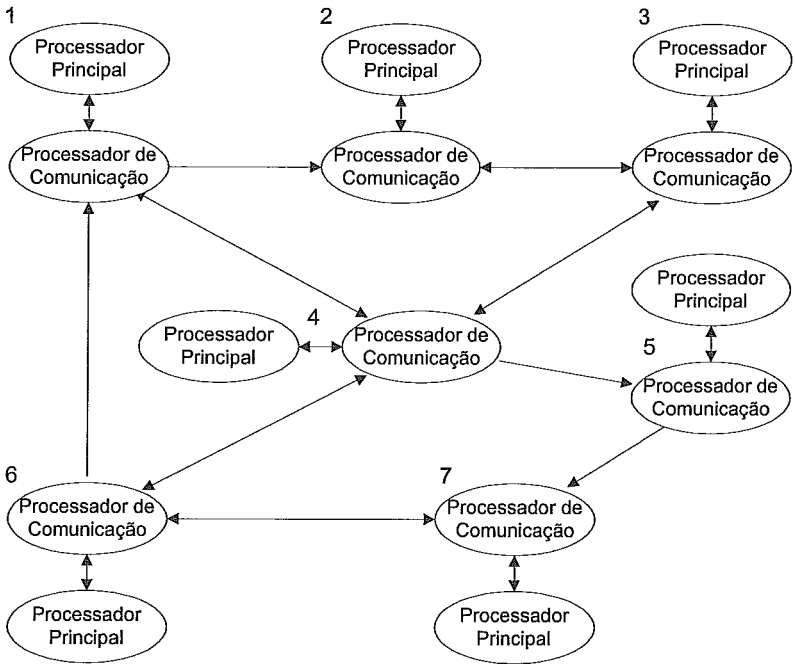

Figura 3.2: Detalhamento da arquitetura do sistema de memória distribuída apresentado na figura 3.1. Cada nó da rede anterior corresponde agora a uma associação de um processador principal e um processador de comunicação.

O fluxo de mensagens na rede caminha através dos canais e dos processadores de comunicação. Estes encaminham as mensagens que recebem para o próprio processador principal do agente ou para outro processador de comunicação da rede, de acordo com o destino da mensagem. Esse repasse de mensagens acontece em cada processador segundo uma função de roteamento predefinida para a rede de comunicação do sistema, de forma que a comunicação seja efetiva e confiável sem que o processador principal tenha qualquer envolvimento no processo de troca de mensagens. Do ponto de vista de uma tarefa, tudo se passa como se houvesse um canal de comunicação entre qualquer par de agentes do sistema.

A flexibilidade de suportar qualquer tipo de comunicação entre os agentes é bastante relevante ao se tratar sistemas distribuídos envolvendo conhecimento. Segundo HALPERN e MOSES (1984), a comunicação em um sistema distribuído pode ser vista como o ato de transformar o estado de conhecimento do sistema. Assim sendo, a evolução da comunicação entre os agentes pode ser entendida como uma ascensão numa hierarquia de estados de conhecimento relativos ao grupo envolvido no sistema, ou seja, a forma como o conhecimento dos fatos é disseminado no grupo à medida em que os agentes se comunicam, desde o conhecimento individual de um agente até a obtenção de conhecimento comum. Um dos resultados importantes acerca de comunicação em

sistemas distribuídos envolvendo conhecimento é que se há falhas na comunicação então não é possível alcançar conhecimento comum. Outros resultados relevantes foram apresentados em FAGIN *et al.* (1995).

3.2 Premissas temporais em sistemas de memória distribuída

Nesta seção o modelo teórico de arquitetura apresentado na seção anterior é revisto no intuito de refletir também as propriedades temporais do sistema considerado. Ao considerar as propriedades temporais primitivas inerentes ao sistema, podemos especificar duas variações admissíveis de sistemas de memória distribuída: assincronismo total e sincronismo total.

Sistema Assíncrono

Um sistema totalmente assíncrono ou simplesmente assíncrono é caracterizado pelas seguintes propriedades:

- Cada nó é dirigido por sua própria base de tempo, denominada relógio local. Esta é uma noção temporal local do agente, e independe de qualquer outro fator do sistema;
- O tempo total de envio de uma mensagem de um nó a outro adjacente através de um dos canais de comunicação da rede é finito porém imprevisível, ou seja: as mensagens chegam em algum momento futuro após serem enviadas, mas não há como prever o atraso na entrega.

No modelo assíncrono, exceto possivelmente no início da execução do algoritmo, a computação em um nó só acontece como conseqüência do recebimento de mensagens. No caso de ocorrer a chegada de mais de uma mensagem ao mesmo tempo, estas são aceitas não deterministicamente. Todos os agentes têm consciência de que após algum tempo finito no futuro de quando foram enviadas as mensagens serão recebidas por seus destinatários.

Nenhuma informação temporal é utilizada que referencie a noção de tempo global do sistema (pois tal noção não existe no modelo assíncrono). A computação em um nó pode ser descrita como:

- Um conjunto de ações a serem tomadas inicialmente (se o nó em questão deve iniciar sua computação e enviar mensagens espontaneamente, ao contrário de fazer isso como conseqüência de ser "despertado" pelo recebimento de alguma mensagem);

- Ações que devem ser tomadas como conseqüência do recebimento de alguma mensagem, quando tal situação acontece sob certas condições booleanas.

O modelo assíncrono está bem próximo das condições reais de funcionamento dos sistemas de memória distribuída em computação, porém suas características temporais acarretam muitas limitações no nível do projeto de algoritmos para essa classe de sistemas.

Sistema Síncrono

As seguintes propriedades temporais descrevem o modelo totalmente síncrono (ou simplesmente síncrono):

- Todos os nós são guiados por uma mesma base de tempo global referida como o relógio global do sistema distribuído, que gera intervalos de tempo de tamanho fixo maior que zero.
- O tempo de entrega de uma mensagem entre nós vizinhos é necessariamente não nulo e estritamente menor que a duração de um intervalo do relógio global.

O início de cada intervalo de tempo é indicado por um "pulso" do relógio global. Para um inteiro $s \geq 0$, o pulso s indica o início do intervalo s. No pulso $s = 0$ os nós que devem iniciar a computação no sistema distribuído enviam mensagens a um subconjunto (possivelmente vazio) de seus canais de comunicação adjacentes. No pulso $s > 0$ todas as mensagens enviadas no pulso $s - 1$ já foram recebidas por seus destinatários (de acordo com a especificação do modelo), e então os nós do sistema podem fazer suas computações e as devidas trocas de mensagens.

Para garantir as propriedades do modelo síncrono assume-se a hipótese de que a computação feita pelos nós durante um intervalo leva tempo zero. Sem essa hipótese, a duração do intervalo poderia não ser suficiente para acomodar a entrega das mensagens mais a computação local.

Assim como no caso assíncrono pode haver no caso síncrono um conjunto de nós aptos a enviar mensagens no pulso $s = 0$, contudo as propriedades do modelo síncrono permitem que os nós façam computações independentemente do recebimento de qualquer mensagem, pois o funcionamento dos agentes é ditado pelo relógio global e não pela chegada de mensagens. Porém, para que a computação global do sistema tenha outro significado que não o do simples paralelismo de tarefas, ao menos uma mensagem

deve ser enviada por pelo menos um nó do sistema durante a execução de um algoritmo distribuído.

Apesar da aparência um tanto distante da realidade do modelo síncrono, ele apresenta um forte apelo no desenvolvimento de algoritmos distribuídos, na medida em que viabiliza o desenvolvimento simplificado de algoritmos que no modelo assíncrono seriam extremamente complexos. Há ainda a possibilidade de aplicar uma transformação mecânica em um algoritmo síncrono para transformá-lo em assíncrono. Outra propriedade interessante que pode ser utilizada no desenvolvimento de algoritmos é que no modelo síncrono os nós podem ganhar informação temporal acerca do estado do sistema apenas esperando, isto é, contanto os pulsos. Isso significa que há troca de informação entre as tarefas mesmo que não haja troca de mensagens.

Levando em conta as propriedades de cada modelo, podemos concluir que todo algoritmo assíncrono é também um algoritmo síncrono, isto é, se um algoritmo foi projetado para o modelo assíncrono e funciona corretamente sob as hipóteses deste modelo, então ele também funcionará corretamente sob as hipóteses do modelo síncrono para a escolha de uma duração de intervalo apropriada (capaz de acomodar as computações do nó). Isso acontece porque as condições em que a comunicação se dá em um modelo síncrono é uma das infinitas possibilidades que o modelo assíncrono permite. O inverso desta implicação (isto é, que o algoritmo síncrono executa corretamente no modelo assíncrono) pode ser atingido com uma transformação apropriada no algoritmo.

3.3 Modelo de algoritmo para uma tarefa de um sistema de memória distribuída

Será descrito nesta seção um modelo genérico de alto nível de algoritmo para um sistema de memória distribuída. A apresentação do modelo permite analisar o procedimento computacional e a comunicação no nível da tarefa, e conseqüentemente desperta a atenção para como a execução de várias instâncias do algoritmo influi no funcionamento como um todo do sistema de memória distribuída. Ficará evidente também o papel crucial que as trocas de mensagens representam nessa classe de sistemas, em específico no controle do fluxo de computação de uma tarefa.

Para simplificar a notação utilizada, iremos representar um algoritmo distribuído por um grafo conexo direcionado $G_T=(N_T, D_T)$, onde o conjunto N_T é um

conjunto de n tarefas e o conjunto de arestas direcionadas D_T é um conjunto de canais unidirecionais de comunicação. Para um tarefa t, definimos In_t como o conjunto de arestas dirigidas a t (arestas de entrada de t), e Out_t o conjunto de arestas direcionadas de t para seus nós vizinhos (arestas de saída de t).

O algoritmo *Tarefa_t* mostrado na figura 3.3 descreve o comportamento de uma tarefa genérica t. Apesar do processo de envio de mensagens estar ao final da computação isso não é necessário; a computação e a troca de mensagens podem estar intercaladas no conjunto de ações seqüenciais da tarefa.

Uma tarefa t é **reativa** ou **regida por mensagens** no sentido que ela normalmente só executa alguma computação (incluindo o envio de mensagens para outras tarefas) em resposta ao recebimento de uma mensagem de outra tarefa. Uma exceção a essa regra é que pelo menos um tarefa deve ser capaz de enviar mensagens espontaneamente no início de sua execução, para indicar o início da computação no sistema às outras tarefas. Além disso, uma tarefa pode inicialmente fazer computações de inicialização.

Algoritmo *Tarefa_t*:

Faz computação inicial;

Envia mensagens para um subconjunto (possivelmente vazio) de Out_t;

Repita

Recebe mensagem em $c_1 \in In_t$ **e** $B_1 \Rightarrow$

- Faz algumas computações;
- **Envia** mensagens para um subconjunto (possivelmente vazio) de Out_t

ou ...

ou

Recebe mensagem em $c_n \in In_t$ **e** $B_n \Rightarrow$

- Faz algumas computações;
- **Envia** mensagens para um subconjunto (possivelmente vazio) de Out_t

Até que a condição global de terminação seja atingida por *t*.

Figura 3.3: Algoritmo *Tarefa_t*

De acordo com o algoritmo a tarefa *t* é executada até que uma condição global de parada seja percebida por *t*. Enquanto isso não acontece, o algoritmo procede fazendo iterações no laço de repetição. A cada iteração, *t* executa alguma computação e pode enviar mensagens. A computação executada por iteração corresponde a um grupo de comandos agrupados sob uma guarda. Guarda é uma condição da forma: "recebe mensagem em $c_k \in In_t$ **e** B_k" para alguma condição booleana B_k, onde $1 \leq k \leq n$ (o termo **recebe** presente na guarda corresponde a uma operação da tarefa para receber mensagens). O laço de repetição contém *n* guardas e seus respectivos subgrupos de comandos (denominados **comando guardados**) agrupados por conectivos **ou**.

Uma guarda é satisfeita quando existe uma mensagem disponível para recebimento imediato no canal c_k e além disso a condição booleana B_k é verdadeira. Essa condição pode ou não depender da mensagem disponível para recebimento. A cada iteração do laço, apenas uma guarda dentre as que estejam satisfeitas deve ser executada. Se nenhuma guarda for satisfeita, a tarefa é suspensa até que alguma se torne

satisfeita. Se mais de uma estiver satisfeita, então uma dentre elas é escolhida arbitrariamente.

Assim como Recebe, o termo Envia presente nos comandos guardados corresponde a uma operação da tarefa de enviar mensagem através dos canais especificados.

O algoritmo apresentado aborda todos os aspectos que devem ser levados em conta ao se desenvolver um sistema distribuído onde a comunicação é totalmente realizada através de troca de mensagens. O modelo genérico pode facilmente ser adaptado ao se reunir as informações sobre as peculiaridades do sistema a ser desenvolvido e as propriedades e limitações da plataforma de implementação escolhida.

3.4 Modelo de computação para sistemas de memória distribuída

Até o momento já estabelecemos arquitetura, primitivas temporais e modelo de algoritmo para os sistema de memória distribuída com comunicação baseada em troca de mensagens. Porém, ainda não temos formalismo adequado para falar de como a computação ocorre nestes sistemas. Vamos agora estabelecer um modelo formal de computação bem detalhado no que se refere à especificação das propriedades locais e globais das computações distribuídas.

3.4.1 Especificação do Modelo

Introduziremos um formalismo baseado em eventos para descrever as computações distribuídas que nos permitirá ser muito mais precisos no tratamento de propriedades globais, principalmente no que se refere a propriedades temporais no caso assíncrono. Todavia, já vimos que o modelo síncrono é muito mais simples de ser descrito e representado por conter uma restrição temporal no que se refere a ocorrência de eventos, podendo ser considerado como um caso particular do assíncrono. Consequentemente o modelo apresentado também pode ser aplicado a ele.

Sistema Distribuído

Um Sistema distribuído é uma coleção finita de agentes interconectados de alguma forma por uma rede de canais de comunicação. A comunicação é alcançada por meio do envio de mensagens através dos canais da rede. Utilizaremos a notação n_i para designar o agente i e o par (n_i, n_j) para designar o canal de comunicação entre os agentes

i e j direcionado de i para j. A comunicação é garantida (não tem falhas) e o tempo de entrega das mensagens é finito porém indeterminado para o caso dos sistemas distribuídos assíncrono.

Protocolo

É um algoritmo distribuído, possivelmente não determinístico, que especifica as ações de cada agente em resposta recebimento de uma mensagem (conforme apresentado na seção 3.3 deste capítulo).

Evento

Um evento é a tupla $\xi=<n_i, t, \varphi, \sigma, \sigma', \Phi>$ onde:

- n_i é o nó onde o evento ocorre;
- t é o tempo dado pelo relógio local de n_i em que o evento acorreu.
- φ é a mensagem, se houver alguma, que disparou o evento com seu recebimento em n_i;
- σ é o estado de n_i anterior à ocorrência do evento;
- σ' é o estado de n_i imediatamente posterior à ocorrência do evento;
- Φ é o conjunto de mensagens, se houver, enviadas por n_i como conseqüência da ocorrência do evento.

Para nosso modelo computacional, o evento é a unidade básica da noção temporal do modelo assíncrono.

Chamamos *Eventos Internos* os eventos que ocorrem sem nenhuma causa imediata externa ($\varphi = 0$).

Nos sistemas onde há trocas de mensagens, os eventos refletem o recebimento ou envio de uma mensagem, o que significa que um evento de envio de mensagem num modelo sem falhas de comunicação necessariamente acarreta um evento de recebimento de mensagem para outro nó.

Computação distribuída (ou execução)

É uma execução de um algoritmo distribuído, ou seja: é a execução em paralelo de todas as tarefas envolvidas em um algoritmo distribuído, em seus respectivos agentes.

Uma computação distribuída pode ser vista como um conjunto de eventos Ξ. Σ_i é a seqüência de estados pelos quais um nó n_i passa conforme a computação Ξ evolui.

O primeiro membro de Σ_i é o estado inicial de n_i e o último, seu estado final. Cada mudança de estado é conseqüência da ocorrência de um evento de Ξ para o nó n_i.

3.4.2 Propriedades temporais e estados de uma computação distribuída

Iremos incorporar as propriedades temporais que se aplicam às computações distribuídas assíncronas estabelecendo ordenações sobre o conjunto de eventos de uma computação (ou execução).

Para estabelecer uma ordem temporal entre a ocorrência dos eventos em uma computação vamos definir a relação $\prec$ a seguir.

Relação temporal $\prec$

Sejam v_1 e v_2 dois eventos. Então $v_1 \prec v_2$ se e somente se :

(i) Ambos v_1 e v_2 ocorrem no mesmo nó, respectivamente nos instantes t_1 e t_2 tais que $t_1 < t_2$. Nenhum evento v' ocorre no mesmo nó no instante t tal que $t_1 < t < t_2$.

(ii) Os eventos v_1 e v_2 ocorrem em nós vizinhos, e existe uma mensagem φ que é enviada em v_1 e recebida em v_2.

O significado da relação binária $\prec$ é "v_1 aconteceu imediatamente antes de v_2", o que só faz sentido quando consideramos eventos em uma mesma execução. Como conseqüência temos que a relação $\prec$ é acíclica.

Relação temporal $\prec^+$

Em função de $\prec$ podemos definir uma segunda relação binária $\prec^+$, tal que $\prec^+$ é o fecho transitivo e irreflexivo de $\prec$. $\prec^+$ estabelece uma ordem parcial no conjunto de eventos Ξ. Dois eventos v_1 e v_2 que não se relacionam em $\prec^+$ tais que:

$$(v_1, v_2) \in \Xi \times \Xi - \prec^+ \text{ e}$$

$$(v_2, v_1) \in \Xi \times \Xi - \prec^+$$

são ditos concorrentes. As relações $\prec$ e $\prec^+$ e podem ser melhor entendidas através de uma representação gráfica dos eventos em uma execução, que será apresentada a seguir.

Grafo de precedência

Uma execução Ξ pode ser visualizada através de um grafo $H = (\Xi, \prec)$, chamado grafo de precedência. Os nós de H são eventos de Ξ, e as arestas direcionadas

representam a relação $\prec$. Em vista da definição da relação $\prec$, o grafo H é direcionado e acíclico.

Apresentamos na figura 3.4 um grafo de precedência com quatro nós. O grafo de precedência permite melhor visualização ao se representar os eventos associados a um mesmo nó em uma linha horizontal, na ordem dada pela relação $\prec$. Nessa representação, as arestas horizontais correspondem a eventos internos para um nó, enquanto as outras arestas relacionam eventos correspondentes de envio e recebimento de mensagens entre dois nós.

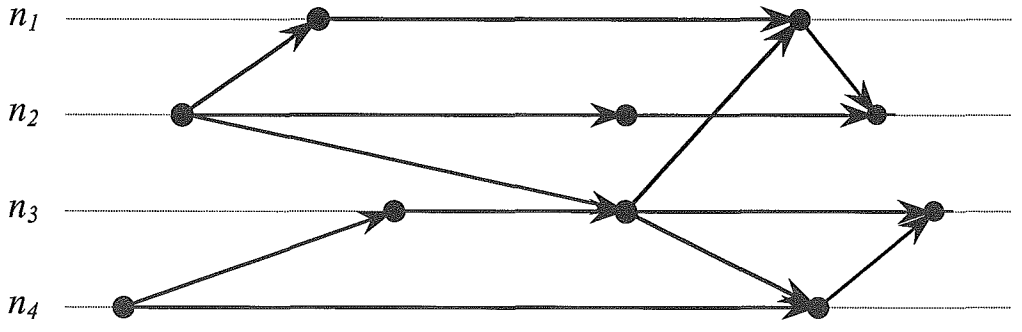

Figura 3.4: Grafo de precedência com quatro nós.

Passado e Futuro

Podemos utilizar a relação $\prec^+$ para definir o passado e o futuro para um evento ξ em relação a uma execução Ξ:

Passado(ξ) = { $\xi' \in \Xi \mid \xi' \prec^+ \xi$ }

Futuro(ξ) = { $\xi' \in \Xi \mid \xi \prec^+ \xi'$ }

Na figura 3.5 podemos observar o grafo de precedência com quatro nós mostrado anteriormente, e o evento ξ indicado. As regiões "cônicas" no grafo de precedência em torno do evento ξ ocorrido para o nó n_3 representam $\{\xi\} \cup$ Passado(ξ) e $\{\xi\} \cup$ Futuro(ξ).

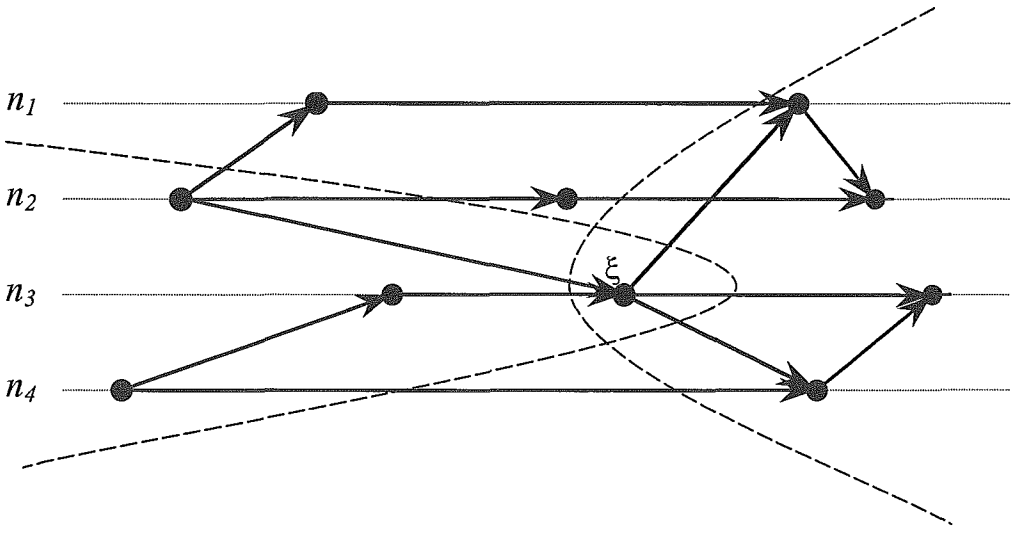

Figura 3.5: A linha tracejada cônica a esquerda delimita a região que compreende {ξ} ∪ Passado(ξ) e a linha tracejada cônica a direta delimita a região de {ξ} ∪ Futuro(ξ).

De posse do formalismo apresentado para descrever as relações de precedência entre eventos de uma computação podemos agora fazer definições que envolvam propriedades temporais e outras propriedades globais do modelo, o que é de especial importância para o caso assíncrono.

Estado do sistema

Um *estado do sistema* é uma coleção de estados locais, um para cada nó e um "estado de aresta" para cada aresta. Mais formalmente, um estado do sistema é definido por:

(i) Para cada agente n_i, o estado local de n_i;

(ii) Para cada canal (n_i, n_j) um conjunto de mensagens representando as mensagens em trânsito de n_i para n_j.

Estado Global Consistente ou Estado Global

O estado global deve refletir o estado local de cada agente e mais o conjunto de mensagens em trânsito. Ele é considerado consistente no sentido de que não existam mensagens do futuro para o passado. Apresentaremos duas definições equivalentes de estado global.

1. Definição em função de ordens totais sobre o conjunto de eventos

Para definir estado global consistente, é necessário estabelecer uma *ordem total* $<$ em Ξ consistente com $\prec^+$.Ξ - 1 pares de eventos consecutivos $(\xi_1, \xi_2) \in <$ podem

ser identificados de forma que todo evento $\xi \neq \xi_1, \xi_2$ é tal que $\xi < \xi_1$ ou $\xi_2 < \xi$. Associado a cada par (ξ_1, ξ_2) de eventos consecutivos em $<$ está um estado do sistema denotado por estado_do_sistema(ξ_1, ξ_2) com as seguintes características:

Para um nó n_i, σ_i é o estado resultante da ocorrência do evento mais recente (com o maior tempo de ocorrência) em n_i, por exemplo ξ, tal que $\xi_1 \not< \xi$ (pode-se considerar inclusive $\xi = \xi_1$).

Para cada aresta (n_i, n_j) Φ_{ij} é o conjunto de mensagens enviadas em conexão com um evento ξ tal que $\xi_1 \not< \xi$ (incluindo a possibilidade de $\xi = \xi_1$) e recebidas em conexão com um evento ξ' tal que $\xi' \not< \xi_2$ (incluindo a possibilidade de $\xi' = \xi_2$).

Um estado Ψ do sistema é global se e somente se: ou todos os nós estão em seu estado inicial e todas as arestas estão vazias, ou todos os nós estão em seu estado final e todas as arestas estão vazias, ou existe uma ordem total $<$ consistente com $\prec^+$ na qual existe um par (ξ_1, ξ_2) de eventos consecutivos tal que Ψ = estado_do_sistema (ξ_1, ξ_2).

2. Definição em função de partições no conjunto de eventos

Outra definição para estado global: Um estado do sistema Ψ é global se e somente se é representado por uma partição (Ξ_1, Ξ_2) de Ξ tal que :

Passado$(\xi) \subseteq \Xi_1$ sempre que $\xi \in \Xi_1$

Ou da mesma forma:

Futuro$(\xi) \subseteq \Xi_2$ sempre que $\xi \in \Xi_2$

As partições que obedecem a esta restrição são chamadas de **cortes consistentes**.

Segundo esta definição, Ψ = estado_do_sistema(Ξ_1, Ξ_2).

Assim sendo, o estado do sistema representado pela partição (Ξ_1, Ξ_2) quando esta é um corte consistente é:

a) Para cada agente n_i, o estado local de n_i é o estado decorrente do evento de Ξ_1 mais recente ocorrendo em n_i;
b) Para cada aresta (n_i, n_j), um conjunto de mensagens enviadas por n_i através de eventos de Ξ_1 e recebidas por n_j através de eventos de Ξ_2.

Podemos representar os cortes consistentes sobre um grafo de precedência, conforme mostrado na figura 3.6.

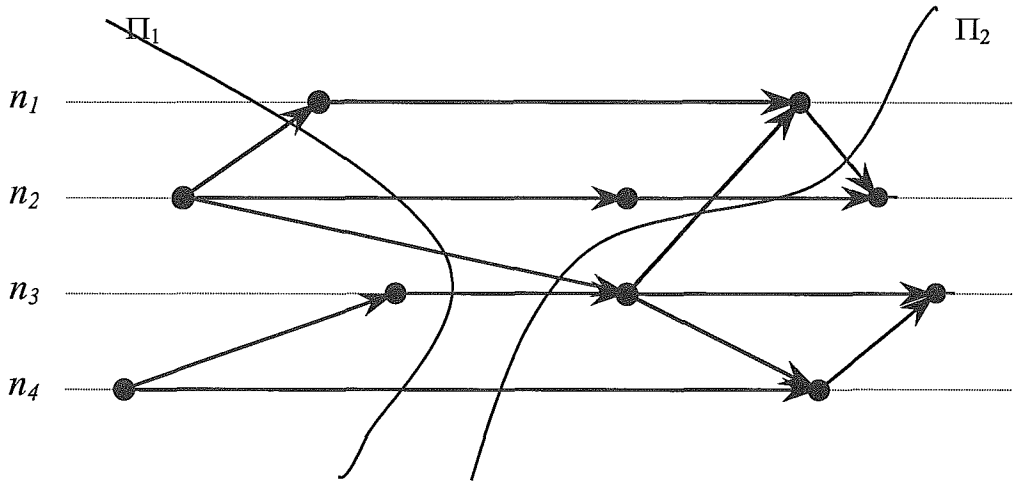

Figura 3.6: Estão representadas sobre o grafo de precedência duas partições Π_1 e Π_2, sendo que a partição Π_1 corresponde a um corte consistente, enquanto na partição Π_2 há uma aresta do futuro para o passado, de forma que esta não pode ser considerada um corte consistente.

Passado e Futuro de um estado global

A partir das definições de estado global consistente podemos estender o conceito de passado e futuro de um evento para passado e futuro em relação a um estado global Ψ:

$$\text{Passado}(\Psi) = \cup_{\xi \in \Xi 1} [\{\xi\} \cup \text{Passado}(\xi)]$$

$$\text{Futuro}(\Psi) = \cup_{\xi \in \Xi 2} [\{\xi\} \cup \text{Futuro}(\xi)]$$

onde Ψ = estado_do_sistema(Ξ_1, Ξ_2).

Dizemos que um estado global Ψ_1, antecede outro estado global Ψ_2 em uma computação Ξ se e somente se:

$\text{Passado}(\Psi_1) \subset \text{Passado}(\Psi_2)$ ou

$\text{Futuro}(\Psi_2) \subset \text{Futuro}(\Psi_1)$.

Visão Passada do agente

Dada uma computação distribuída Ξ, a visão passada do agente n_j em relação ao estado global s representado pela partição (Ξ_1, Ξ_2) é o conjunto de eventos que ocorrem para o agente n_j no passado de ξ_j, onde ξ_j é o evento associado ao estado local do agente n_j, ou seja, ξ_j é o evento de Ξ_1 mais recente ocorrendo para o agente n_j no estado global s:

$$\text{Vp}(n_j, s) = \{\xi_j' \in \Xi_1 \mid \xi_j' \prec^+ \xi_j\}$$

Visão Futura do agente

Dada uma computação distribuída Ξ, a visão Futura do agente n_j em relação ao estado global s representado pela partição (Ξ_1, Ξ_2) é o conjunto de eventos que ocorrem para o agente n_j no futuro de ξ_j, onde ξ_j é o evento associado ao estado local do agente n_j, ou seja, ξ_j é o evento de Ξ_1 mais recente ocorrendo para o agente n_j no estado global s:

$$Vf(n_j, s) = \{\xi_j' \in \Xi_2 \mid \xi_j \prec^+ \xi_j'\}$$

Concluímos agora a formalização do modelo completo para sistemas distribuídos que será utilizado neste trabalho. Para ilustrar a aplicabilidade do modelo, vejamos como exemplo um sistema com três agentes rodando o algoritmo distribuído para propagação da informação com realimentação (PIF).

Exemplo: Algoritmo de Propagação da Informação com Realimentação (PIF)

O algoritmo distribuído para PIF é bastante simples e ilustra bem os conceitos de execuções e cortes consistentes. Ele funciona basicamente da seguinte forma: o agente iniciador manda a mensagem φ a todos os outros agentes do sistema. Cada agente, ao receber φ, manda φ para todos os vizinhos exceto aquele do qual recebeu φ.

Consideremos um sistema com três agentes rodando PIF onde fixamos o iniciador do algoritmo como sendo o agente a_1, conforme apresentado em COSTA (2001).

Primeiramente definimos o conjunto $E = \{ e_1, e_2, e_3, e_4, e_5, e_6, e_7 \}$ de eventos que caracterizam o envio de mensagens entre os agentes (os eventos têm o mesmo significado independente da execução). Cada conjunto E_m de eventos possíveis para o agente m é dado por:

$$E_1 = \{ e_1, e_6, e_7 \}, E_2 = \{ e_2, e_5 \}, E_3 = \{ e_3, e_4 \}$$

Fixando a_1 como iniciador, obtemos seis execuções possíveis r_1, r_2, r_3, r_4, r_5 e r_6, conforme a figura 3.7. As execuções resultam das possíveis ordenações dos acontecimentos dos eventos para cada agente.

Enumeramos os possíveis estados globais ou cortes consistentes que ocorrem nas seis execuções.

c_0: $V_1 = \{ \}$; $V_2 = \{e_1, e_2, e_3, e_4, e_5, e_6, e_7\}$

c_1: $V_1 = \{e_1\}$; $V_2 = \{e_2, e_3, e_4, e_5, e_6, e_7\}$

c_2: $V_1 = \{e_1,e_2\}$; $V_2 = \{e_3, e_4, e_5, e_6, e_7\}$

c_3: $V_1 = \{e_1,e_2,e_3\}$; $V_2 = \{e_4, e_5, e_6, e_7\}$

c_4: $V_1 = \{e_1,e_2,e_3,e_4\}$; $V_2 = \{e_5, e_6, e_7\}$

c_5: $V_1 = \{e_1,e_2,e_3,e_4,e_5\}$; $V_2 = \{e_6, e_7\}$

c_6: $V_1 = \{e_1,e_2,e_3,e_4,e_5,e_6\}$; $V_2 = \{e_7\}$

c_7: $V_1 = \{e_1,e_2,e_3,e_4,e_5,e_6,e_7\}$; $V_2 = \{\}$

c_8: $V_1 = \{e_1,e_2,e_3,e_6\}$; $V_2 = \{e_4, e_5, e_7\}$

c_9: $V_1 = \{e_1,e_2,e_3,e_6,e_4\}$; $V_2 = \{e_5, e_7\}$

c_{10}: $V_1 = \{e_1,e_2,e_3,e_4,e_5,e_7\}$; $V_2 = \{e_6\}$

c_{11}: $V_1 = \{e_1,e_4\}$; $V_2 = \{e_2, e_3, e_5, e_6, e_7\}$

c_{12}: $V_1 = \{e_1,e_2,e_4,e_5\}$; $V_2 = \{e_3, e_6, e_7\}$

c_{13}: $V_1 = \{e_1,e_2,e_4\}$; $V_2 = \{e_3, e_5, e_6, e_7\}$

c_{14}: $V_1 = \{e_1,e_2,e_4,e_5,e_7\}$; $V_2 = \{e_3, e_6\}$

c_{15}: $V_1 = \{e_1,e_4,e_5\}$; $V_2 = \{e_2, e_3, e_6, e_7\}$

c_{16}: $V_1 = \{e_1,e_4,e_5,e_7\}$; $V_2 = \{e_2, e_3, e_6\}$

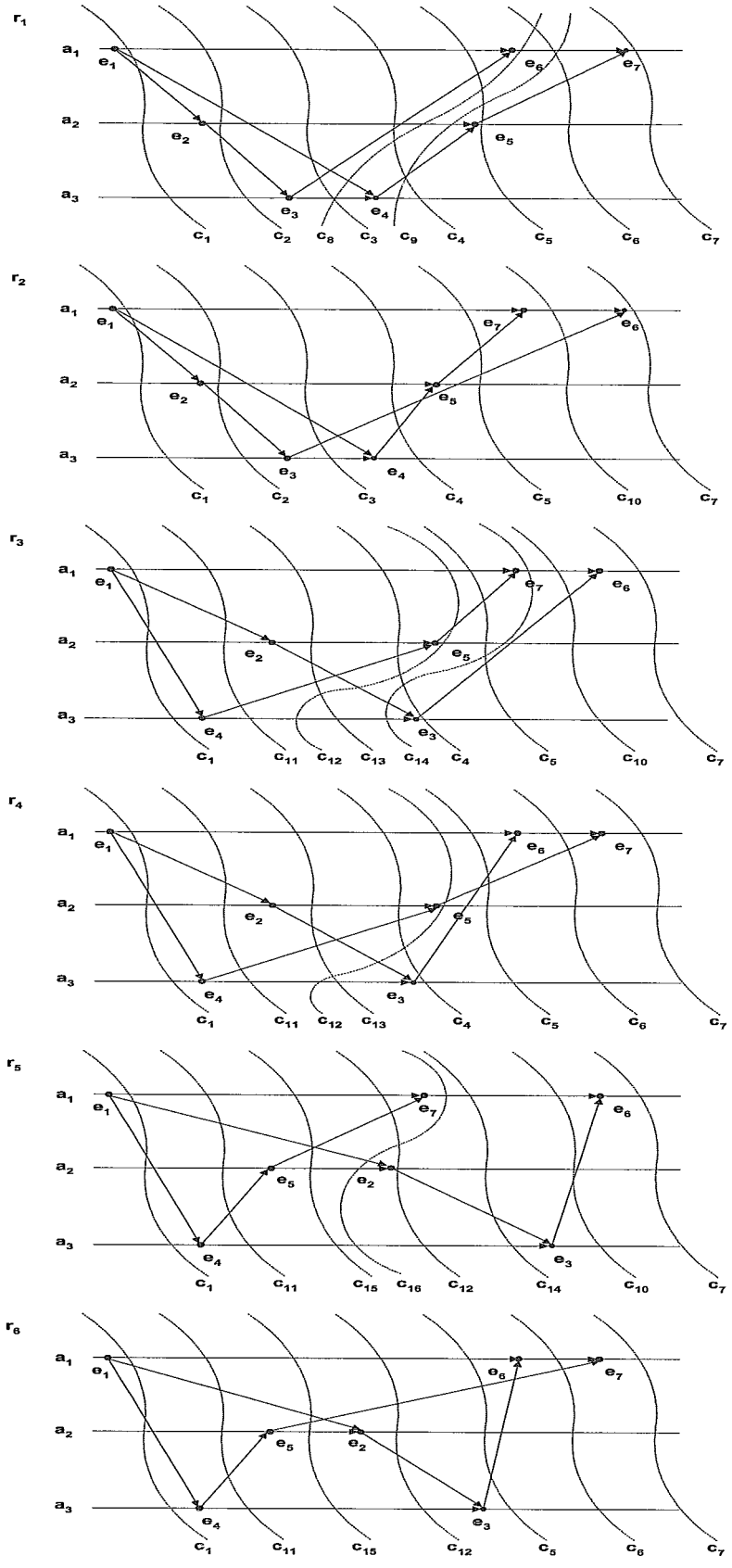

Figura 3.7: Grafo de precedência para as possíveis execuções de PIF em um sistema de três agentes, com os respectivos cortes consistentes.

Em função dos cortes, temos os seguintes conjuntos de estados ou visões passadas para cada agente:

- Agente a_1

$Vp(a_1,c_i) = \{e_1\}$ $\quad i = 1, 2, 3, 4, 5, 11, 12, 13, 15.$

$Vp(a_1,c_j) = \{e_1, e_6\}$ $\quad j = 6, 8, 9.$

$Vp(a_1,c_k) = \{e_1, e_7\}$ $\quad k = 10, 14, 16.$

$Vp(a_1,c_l) = \{e_1, e_6, e_7\}$ $\quad l = 7.$

- Agente a_2

$Vp(a_2,c_i) = \{\}$ $\quad i = 1, 11.$

$Vp(a_2,c_j) = \{e_2\}$ $\quad j = 2, 3, 4, 8, 9, 13.$

$Vp(a_2,c_k) = \{e_5\}$ $\quad k = 15, 16.$

$Vp(a_2,c_l) = \{e_2, e_5\}$ $\quad l = 5, 6, 7, 10, 12, 14.$

- Agente a_3

$Vp(a_3,c_i) = \{\}$ $\quad i = 1, 2.$

$Vp(a_3,c_j) = \{e_3\}$ $\quad j = 3, 8.$

$Vp(a_3,c_k) = \{e_4\}$ $\quad k = 11, 12, 13, 14, 15, 16.$

$Vp(a_3,c_l) = \{e_3, e_4\}$ $\quad l = 4, 5, 6, 7, 10.$

Apesar do modelo apresentado ser teórico e abstrato, as hipóteses assumidas são bastante coerentes com o modelo real de sistemas distribuídos utilizado atualmente. Caso estivéssemos interessados apenas em sistemas síncronos não seria necessário conceber modelos onde as relações temporais são parciais, pois em sistemas síncronos podemos descrever relações temporais totais uma vez que os eventos do sistema ocorrem em pulsos em que a precedência no tempo é bem determinada. Também é bem mais fácil definir estados globais em sistemas síncronos. Basta considerá-los como uma função direta dos estados locais de cada agente em um dado pulso de tempo.

Uma vez definidos os estados globais podemos facilmente estabelecer estruturas Kripke sobre sistemas distribuídos síncronos, como foi feito no capítulo 2. Para definir estruturas Kripke sobre modelos assíncronos teremos que utilizar as definições de estados globais e a ordenação temporal parcial baseada em eventos apresentadas neste capítulo, o que torna o modelo mais complexo. Como vimos, a

formalização do conhecimento em sistemas distribuídos que buscamos envolve os estados de conhecimento individuais de cada agente nos diversos estados globais da execução de um algoritmo, e as relações entre esses estados. Consequentemente o formalismo para tratar estados globais em sistemas assíncronos será de vital importância para os próximos capítulos.

Capítulo 4

Lógica Dinâmica de Conhecimento para Sistemas Síncronos

Apresentamos no capítulo 2 a lógica modal L_m para representar o conhecimento em um grupo de agentes. A lógica L_m consegue representar o conjunto dos estados epistêmicos dos agentes do sistema, de forma estática. A cada nova informação percebida pelos agentes é preciso reformular os estados epistêmicos, e conseqüentemente redefinir as relações R_i.

Para poder representar o conhecimento evoluindo em um sistema distribuído, apresentamos também no capítulo 2 a lógica modal de conhecimento e tempo para sistemas distribuídos com múltiplos agentes proposta por Daniel Lehmann em LEHMANN (1984). Tal linguagem é capaz de expressar os conceitos de conhecimento de um agente e do grupo em estados pontuais temporalmente ordenados. O trabalho apresenta um sistema axiomático completo para conhecimento comum e a uma caracterização precisa do papel do tempo na evolução do conhecimento.

Observando porém a maioria dos artigos em sistemas distribuídos duas noções básicas aparecem repetidamente: estados e ações. Conforme visto no capítulo 3, as tarefas do modelo apresentado para sistemas de memória distribuída são reativas, ou seja, executam suas computações em resposta aos eventos provocados por ações de envido de mensagens (com possível exceção da tarefa iniciadora). Considere um algoritmo distribuído muito simples, constituído de um único processo rodando um programa seqüencial. Como Lamport aponta em LAMPORT (1985), uma execução deste algoritmo pode ser vista como uma seqüência intercalada de estados e ações, conforme ilustrado na figura 4.1. Sob esta ótica, um processo é um autômato que está sempre em um estado de um conjunto de estados internos possivelmente infinito.

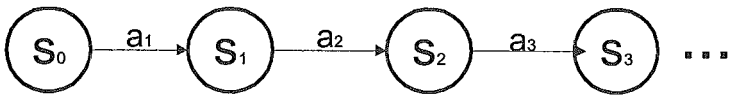

Figura 4.1: Algoritmo distribuído constituído de um único processo rodando um programa seqüencial. S_0, S_1, S_2 e S_3 são estados e a_0, a_1, a_2 e a_3 são ações.

A intenção deste capítulo é propor uma lógica modal para modelar conhecimento e tempo em sistemas síncronos que tenha os conceitos de estados e ações condizentes com o modelo para sistemas de memória distribuída apresentado no capítulo 3. Diferentemente do mecanismo apresentado em LEHMANN (1984) de navegar pelos estados olhando para o futuro, na lógica que iremos apresentar o mecanismo que permite modelar as mudanças de estados é a realização de ações sobre o sistema. A execução de uma ação que faz com que o estado epistêmico do sistema mude e um novo estado possa ser definido para o mundo possível resultante da execução da ação em questão sobre o mundo possível anterior. Chamaremos essa lógica modal de conhecimento e ações de Lógica Dinâmica de Conhecimento, que foi objeto de um artigo apresentado no XXI Congresso da Sociedade Brasileira de Computação - Encontro Nacional de Inteligência Artificial: DELGADO e BENEVIDES (2001). Como exemplo, aplicamos LDC para modelar um problema clássico de conhecimento em sistemas distribuídos: o problema das crianças com lama na testa, que será detalhadamente apresentado.

4.1 Lógica Dinâmica de Conhecimento

Iremos agora propor uma Lógica Dinâmica de Conhecimento (LDC) capaz de representar o conhecimento em um grupo de agentes e incorporar ações sobre os estados epistêmicos do sistema e as mudanças nos estados que advém destas ações. A lógica possui um *frame* bastante flexível, de forma que as restrições de aplicação das ações devem ser representadas como axiomas para cada aplicação, como será feito na seção 4.3 para o problema das crianças com lama na testa.

4.1.1 Sintaxe para Lógica Dinâmica de Conhecimento

A linguagem utilizada é a da lógica modal de conhecimento e crença para um número m de agentes (L_m) apresentada no Capítulo 2 e acrescida das modalidades de ação [$send_i$] e [tick].

Símbolos

Os símbolos da linguagem LDC são:

1. Um conjunto Φ enumerável de símbolos proposicionais;
2. Pontuação: "(" e ")";
3. Conectivos: "$\neg$", "$\wedge$", "$\vee$" e "$\rightarrow$";
4. Modais: K_i, i = 1, 2, 3, ..., m (um para cada agente);
5. Modais: B_i, i = 1, 2, 3, ..., m (um para cada agente);
6. Modal: $[send_i]$, i = 1, 2, 3, ..., m (um para cada agente);
7. Modal: [tick].

Operadores

Os operadores de LDC são definidos da seguinte forma:

1. Os conectivos seguem as definições da lógica proposicional;
2. O operadores modais "K_i" e "B_i" seguem as definições da lógica de conhecimento L_m;
3. O operador modal de ação "$[send_i]$" representa a ação de anúncio por parte do agente i para o grupo de agentes do sistema de que o agente i sabe que está com lama na própria testa, ou seja, $[send_i]\varphi$ indica que φ vale no mundo possível atingido devido à execução da ação em que o agente i anunciou para todos os agentes do grupo que sabe que está com a testa suja, para i=1, 2, 3, ..., m;
4. O operador modal de ação "[tick]" representa a ocorrência de um pulso no relógio global do sistema, ou seja, $[tick]\varphi$ indica que φ vale no mundo possível atingido decorrente da execução de um pulso no relógio global do sistema.

Fórmulas

As fórmulas da linguagem são descritas pelas seguintes regras:

1. Todo símbolo proposicional de Φ é uma fórmula, chamada fórmula atômica.
2. Se φ é uma fórmula, então $(\neg\varphi)$ também é uma fórmula.
3. Se φ e ψ são fórmulas, então $(\varphi \wedge \psi)$, $(\varphi \vee \psi)$ e $(\varphi \rightarrow \psi)$ também são fórmulas.
4. Se φ é uma fórmula, então $K_i\varphi$, também é uma fórmula para i=1, 2, 3, ..., m.
5. Se φ é uma fórmula, então $B_i\varphi$, também é uma fórmula, para i=1, 2, 3, ..., m.

6. Se φ é uma fórmula, então $[\alpha]\varphi$, também é uma fórmula, para toda modalidade de ação $[\alpha] = [tick], [send_i]$, para i=1, 2, ..., m.
7. Nada é uma fórmula, a não ser que seja forçado por um dos itens acima.

4.1.2 Semântica para LDC

A semântica adotada reflete a mesma da lógica L_m para as modalidades de conhecimento e inclui a semântica para as modalidades de ação.

LDC-frame

Um LDC-*frame* $F_{LDC} = (S, R_i, R_{sendi}, R_{tick})$, i=1, 2, 3, ..., m é uma estrutura onde:

- S é um conjunto de *estados* ou *mundos possíveis*;
- R_i, R_{sendi}, R_{tick} são relações binárias em S, ou seja, um conjunto de pares de elementos de S onde i=1, 2, 3, ..., m ($R_i \subseteq S \times S$, $R_{sendi} \subseteq S \times S$, $R_{tick} \subseteq S \times S$);
- A relação R_i é reflexiva, simétrica e transitiva.

Modelo

Um modelo M sobre $F_{LDC} = (S, R_i, R_{sendi}, R_{tick})$ é um par $M = (F_{LDC}, \pi)$, onde π é uma interpretação (função de valoração) que associa valores verdade às primitivas de Φ em cada estado de S, isto é, $\pi: \Phi \times S \rightarrow \{V, F\}$.

Satisfatibilidade

Uma fórmula φ de LDC é verdadeira em (M, s), ou seja, é verdadeira em um estado $s \in S$ para um modelo M quando:

1. $M, s \models p$ se e somente se $\pi(s, p) = V$, onde $p \in \Phi$;
2. $M, s \models \neg\varphi$ se e somente se não é o caso que $M, s \models \varphi$;
3. $M, s \models \varphi \wedge \psi$ se e somente se $M, s \models \varphi$ e $M, s \models \psi$;
4. $M, s \models K_i\varphi$ se e somente se para todo $t \in S$ tal que $(s, t) \in R_i$ temos que $M, t \models \varphi$;
5. $M, s \models B_i\varphi$ se e somente se existe t tal que $t \in S$ e $(s, t) \in R_i$, para o qual temos que $M, t \models \varphi$;

6. M, s $\models [\alpha]\varphi$ se e somente se para todo $t \in S$ tal que $(s, t) \in R_\alpha$ temos que M, t $\models \varphi$, para toda modalidade de ação $[\alpha] = [tick], [send_i]$ para i=1, 2, ..., m.

Seja M = (F, π) um modelo para F_{LDC}. Dizemos que M satisfaz φ se existe algum mundo $s \in S$ tal que M, s $\models \varphi$. Dizemos que φ é satisfatível se existe algum modelo que o satisfaça, caso contrário, dizemos que φ é insatisfatível. Uma fórmula φ é válida em um Modelo M para F_{LDC} se φ é satisfeita em todos os estados de M (para todo $s \in S$, M, s $\models \varphi$). Uma fórmula φ é válida em F_{LDC} se φ é válida em todos os modelos sobre F_{LDC} (para todos M e s, M, s $\models \varphi$).

4.1.3 Sistemas axiomáticos de LDC

Axiomas

- Todas as tautologias do cálculo proposicional;
- $K_i(\varphi \rightarrow \psi) \rightarrow (K_i\varphi \rightarrow K_i\psi)$, para todo φ e $\psi \in$ LDC e i=1, 2, ..., m;
- $[\alpha](\varphi \rightarrow \psi) \rightarrow ([\alpha]\varphi \rightarrow [\alpha]\psi)$, para toda modalidade de ação $[\alpha] = [tick], [send_i]$, para todo φ e $\psi \in$ LDC e i=1, 2, ..., m;
- $B_i\varphi \leftrightarrow \neg K_i \neg \varphi$, para i=1, 2, 3, ..., m (axioma dual);
- $K_i\varphi \rightarrow \varphi$ (axioma do conhecimento);
- $K_i\varphi \rightarrow K_iK_i\varphi$ (introspecção positiva);
- $\neg K_i\varphi \rightarrow K_i\neg K_i\varphi$ (introspecção negativa);

Regras de inferência

- De φ e $\varphi \rightarrow \psi$ derive ψ (*modus ponens*)
- De $\models \varphi$ derive $K_i\varphi$ (generalização do conhecimento).
- De $\models \varphi$ derive $[\alpha]\varphi$, para toda modalidade de ação $[\alpha] = [tick], [send_i]$, para i=1, 2, ..., m (generalização das demais modalidades).

Axiomas e regras de inferência para conhecimento de um grupo de agentes

Considerando a adição dos operadores D_G, E_G e C_G na linguagem, acrescentam-se os seguintes axiomas e regras a LDC:

- D1. $K_i\varphi \rightarrow D_G\varphi$
- D2. $D_G\varphi \rightarrow D_{G'}\varphi$ se $G \subseteq G'$

- C1. $E_G\varphi \rightarrow \wedge_{i \in G} K_i\varphi$
- C2. $C_G\varphi \rightarrow E_G(\varphi \wedge C_G\varphi)$
- H1. $C_G\varphi \rightarrow E_G{}^k\varphi \rightarrow E_G\varphi \rightarrow A_G\varphi \rightarrow D_G\varphi \rightarrow \varphi$ (hierarquia de estados de conhecimento)
- RC1. De $\models \varphi \rightarrow E_G(\psi \wedge \varphi)$ derive $\models \varphi \rightarrow C_G\psi$ (regra da indução).

Teorema: S_{LDC} é uma axiomatização correta e completa para LDC (a prova encontra-se do Anexo B deste trabalho).

4.2 Representação do conhecimento no Problema das Crianças com Lama na Testa

Nesta seção apresentaremos o problema das crianças com lama na testa e a representação do conhecimento no sistema constituído pelos agentes do grupo nele envolvido, utilizando a linguagem L_m apresentada no capítulo 2. Posteriormente, apresentaremos a representação do problema através de LDC.

O problema das Crianças com Lama na testa é um exemplo clássico de sistema baseado em conhecimento que visa focar o conhecimento em um grupo de agentes inteligentes que interagem entre si e podem raciocinar sobre o mundo em que estão. O conhecimento evolui de conhecimento distribuído a conhecimento comum, conforme os agentes vão passando por estados de conhecimento evolutivos no decorrer do tempo. A forma mais conhecida do problema é síncrona, que será vista a seguir.

4.2.1 O problema das crianças com lama na testa

4.2.1.1 Definição do problema

Considere **n** crianças brincando juntas. O pai das crianças disse a elas para que não se sujassem, ou seriam castigadas. Então, todas as crianças tentam se manter limpas, mas adorariam que as outras se sujassem. Acontece que durante a brincadeira algumas crianças se sujam na testa (**k** se sujam, para **n>=k**). Cada uma pode ver a lama na testa das outras crianças, mas não na sua própria testa, então ninguém fala nada. Quando o pai das crianças chega ao local da brincadeira constata que algumas crianças se sujaram. Então ele reúne todas as crianças e avisa: "Pelo menos um de vocês está com a testa suja." A partir de então, o pai pergunta repetidas vezes, pausadamente: "Alguém sabe que está com lama na sua própria testa?"

Assumindo que todas as crianças são inteligentes, que não mentem, têm boa percepção, que não se comunicam durante toda a seqüência de perguntas do pai, e que todas respondem ao mesmo tempo, o que vai acontecer?

4.2.1.2 Resolução

Todas as crianças responderão "Não" k-1 vezes à pergunta feita pelo pai, e na k-ésima pergunta as crianças que estão com a testa suja responderão "Sim".

A prova é por indução no número de crianças sujas, k. Para k=1, o resultado é óbvio: a criança com lama na testa vê que nenhuma outra está suja. Como ela recebe a informação de que pelo menos uma está com a testa suja, então ela conclui que só pode ser ela mesma.

Agora para compreender a intuição por trás do argumento suponha que k=2. Nesse caso há duas crianças com a testa suja, *A* e *B*. Ambas respondem não à primeira pergunta, pois cada uma vê a lama na testa da outra e considera a hipótese onde apenas a outra criança está suja. Considerando o raciocínio da criança *A*, quando a criança *B* também responde não à primeira pergunta, a criança *A* chega a conclusão de que ela também está suja, senão *B* já saberia que estava suja e teria respondido "Sim" à primeira pergunta. A criança *B* segue um raciocínio análogo, e também responde "Sim" à segunda pergunta.

Considere agora k=3. *A*, *B* e *C* estão sujos. *A* raciocina da seguinte forma: "Suponho que eu não estou sujo. Nesse caso, *B* vê apenas *C* sujo, e se *C* não responder sim à primeira pergunta, *B* constatará que está sujo, assim como *C* por um raciocínio análogo, e ambos responderão sim à segunda pergunta" (como no caso k=2). Como *B* e *C* respondem "Não" à segunda pergunta, a criança *A* conclui que isso só acontece porque *B* e *C* vêem mais alguém sujo, além de um ao outro. Logo, *A* conclui que também está sujo, e responde "Sim" à terceira pergunta. O mesmo ocorre para *B* e *C*.

O argumento no caso geral se dá da mesma forma. Suponha que o argumento é válido para o problema com k-1 crianças sujas, e portanto na (k-1)-ésima pergunta todos respondem "Sim". Aplicando a intuição acima para o problema com k crianças sujas, cada criança suja monta uma linha de raciocínio onde para ela são indistinguíveis as situações onde há penas k-1 crianças sujas e ela está limpa, e a situação onde há k crianças sujas, incluindo ela própria. Acontece que, pela hipótese de indução, após a (k-1)- ésima pergunta do pai, se fosse o caso de que houvessem apenas k-1 crianças sujas, as crianças sujas deveriam responder "Sim". Como na (k-1)-ésima

pergunta todos respondem "Não", então cada criança suja conclui que não é o caso que há apenas **k**-1 crianças sujas, e conseqüentemente, como ela não pode ver mais ninguém além das **k**-1 sujas, pode concluir que está suja, e na **k**-ésima pergunta todas as crianças sujas respondem "Sim".

4.2.1.3 Considerações gerais

Veremos agora outros aspectos do problema, para que possamos ter um entendimento completo de todas as informações envolvidas relevantes para sua representação.

Anúncio inicial do pai das crianças

A princípio, para **k** >1, pode parecer que o pai não fornece nenhuma informação adicional com o anúncio de que há pelo menos uma criança com a testa suja. Mas considere o caso em que há duas crianças sujas, *A* e *B*: certamente cada uma sabe que há pelo menos uma criança suja, porém *A* considera possível que sua testa esteja limpa, e nesse caso B não veria nenhuma criança suja. Ou seja: *A* pode considerar o fato de que *B* não saiba que há pelo menos uma criança suja, mesmo que *B* efetivamente o saiba. Quando o pai anuncia esse fato, ele se torna de conhecimento comum do grupo, o que é a base de todo o raciocínio das crianças.

Raciocínio das crianças que não estão sujas

Como o problema envolve **n** crianças, além das **k** crianças sujas participam do sistema **n** - **k** crianças limpas. Como cada criança limpa vê exatamente **k** crianças sujas, cada uma delas monta uma linha de raciocínio análogo ao das crianças sujas, porém aplicado a um número **k** de crianças sujas. Na **k**-ésima pergunta do pai, as crianças limpas ainda não sabem dizer se estão ou não com a testa suja (a certeza sobre a sujeira em suas testas só poderia ser atingida com a (**k**+1)-ésima pergunta), portanto respondem "Não" à pergunta do pai. As crianças sujas, porém, já chegaram à conclusão sobre a sujeira em suas próprias testas, e respondem "Sim". Nesse ponto, as crianças limpas podem concluir que estão com a testa limpa.

Obtenção de conhecimento comum no problema das crianças com a testa suja

Devido à forma como se dá o raciocínio das crianças, após a **k**-ésima pergunta do pai, além de concluir sobre a sujeira em suas testas, o subgrupo formado pelas **k** crianças sujas atinge o conhecimento comum de que elas e somente elas estão com a testa suja. As crianças limpas só são capazes de chegar ao conhecimento comum

de que apenas as k crianças estão sujas após ouvir a resposta "Sim" dessas crianças à k-ésima pergunta.

Características dos agentes no sistema

Um dos fatos que torna possível que as crianças cheguem a qualquer conclusão são as afirmações que fizemos logo a princípio, de que as crianças são inteligentes, todas ouvem o que o pai diz e que não mentem. Se alguma criança considerasse possível que outra pudesse mentir, então nenhuma das crianças poderia chegar às conclusões obtidas. Outro fato importante é a garantia de que todos os agentes sabem abstrair todo o conhecimento contido em qualquer informação do sistema, uma vez que o agente tenha essa informação (agentes oniscientes). Isso é normalmente o que vale nos sistemas de agentes computacionais.

4.2.2 Representação do problema das crianças com lama na testa utilizando a linguagem L_m

Podemos modelar o conhecimento envolvido no sistema utilizando L_m para representar os estados epistêmicos dos agentes.

Consideramos que as n crianças são os agentes do nosso sistema. O pai das crianças não precisa ser representado por um agente, como ficará evidente logo mais.

4.2.2.1 Situação inicial

Primeiramente, numeremos as n crianças de 1 a n, para identificá-las. Consideremos a situação imediatamente anterior ao reencontro do pai com as crianças. Podemos representar os mundos possíveis do sistema como n-tuplas de $(x_1, x_2, x_3..., x_n)$, onde $x_i=0$ significa "a criança i está limpa" e $x_i=1$ significa a criança i está suja. Então, se n=3, a tupla (1, 0, 1) significa: as crianças 1 e 3 estão sujas, e dois está limpa. Nessa situação, consideremos o raciocínio da criança 1: ela vê 3 suja e 2 limpa, então considera dois mundos possíveis (0,0,1) e (1, 0, 1), ou seja: o mundo onde a criança 3 está suja, a criança 2 está limpa e ela está limpa, e o mundo onde a criança 3 está suja, a criança 2 está limpa e ela está suja. Na verdade, a criança i considerará possíveis apenas os mundos que diferirem acerca de x_i.

A representação completa do problema tem 2^n mundos possíveis, cada um representando uma n-tupla de zeros e uns, tal que dois mundos se relacionam do ponto de vista do conhecimento do agente i se eles diferem apenas no i-ésimo componente da n-tupla. O grafo para esse problema é um cubo n-dimensional (FAGIN *et al.*, 1995).

Seja S_i o fato "a criança i está suja"; nesse caso, $\neg S_i$ representa o fato: "a criança i está limpa", para i=1, 2, 3.

Para a linguagem L_m definida e para n=3 crianças, temos as seguintes estruturas:

- O *frame* F = (S, R_i), i=1, 2, 3; onde:

1. S é o conjunto dos mundos possíveis:

S={ (0, 0, 0) ,
(1, 0, 0), (0, 1, 0), (0, 0, 1)
(1, 1, 0), (1, 0, 1), (0, 1, 1)
(1, 1, 1) }

2. As relações Ri são:

R_1 = { [(0, 0, 0), (0, 0 ,0)], [(1, 1, 1), (1, 1 ,1)],
[(1, 0, 0), (1, 0 ,0)], [(0, 1, 0), (0, 1 ,0)], [(0, 0, 1), (0, 0 ,1)]
[(1, 1, 0), (1, 1 ,0)], [(0, 1, 1), (0, 1 ,1)], [(1, 0, 1), (1, 0 ,1)]
[(0, 0, 0), (1, 0 ,0)], [(1, 0 ,0), (0, 0, 0)],
[(0, 1, 1), (1, 1 ,1)], [(1, 1 ,1), (0, 1, 1)],
[(0, 1, 0), (1, 1 ,0)], [(1, 1 ,0) , (0, 1, 0)],
[(1, 0, 1), (0, 0 ,1)], [(0, 0 ,1) , (1, 0, 1)]}

R_2 = { [(0, 0, 0), (0, 0 ,0)], [(1, 1, 1), (1, 1 ,1)],
[(1, 0, 0), (1, 0 ,0)], [(0, 1, 0), (0, 1 ,0)], [(0, 0, 1), (0, 0 ,1)]
[(1, 1, 0), (1, 1 ,0)], [(0, 1, 1), (0, 1 ,1)], [(1, 0, 1), (1, 0 ,1)]
[(0, 0, 0), (0, 1 ,0)], [(0, 1 ,0), (0, 0, 0)],
[(0, 1, 1), (0, 0 ,1)], [(0, 0 ,1), (0, 1, 1)],
[(1, 0, 0), (1, 1 ,0)], [(1, 1, 0), (1, 0 ,0)],
[(1, 0, 1), (1, 1 ,1)], [(1, 1, 1) , (1, 0, 1)]}

R_3 = { [(0, 0, 0), (0, 0 ,0)], [(1, 1, 1), (1, 1 ,1)],
[(1, 0, 0), (1, 0 ,0)], [(0, 1, 0), (0, 1 ,0)], [(0, 0, 1), (0, 0 ,1)]
[(1, 1, 0), (1, 1 ,0)], [(0, 1, 1), (0, 1 ,1)], [(1, 0, 1), (1, 0 ,1)]
[(0, 0, 0), (0, 0 ,1)], [(0, 0 ,0), (0, 0, 1)],
[(1, 1, 1), (1, 1 ,0)], [(1, 1 ,0), (1, 1, 1)],
[(0, 1, 1), (0, 1 ,0)], [(0, 1 ,0) , (0, 1, 1)],
[(1, 0, 1), (1, 0 ,0)], [(1, 0 ,0) , (1, 0, 1)]}

- O modelo M = (F, π), onde:

$\pi(1, 0, 0)(S_1) = \pi(1, 1, 0)(S_1) = \pi(1, 0, 1)(S_1) = \pi(1, 1, 1)(S_1) = V$

$\pi(0, 0, 0)(S_1) = \pi(0, 1, 0)(S_1) = \pi(0, 0, 1)(S_1) = \pi(0, 1, 1)(S_1) = F$

$\pi(0, 1, 0)(S_2) = \pi(1, 1, 0)(S_2) = \pi(0, 1, 1)(S_2) = \pi(1, 1, 1)(S_2) = V$

$\pi(0, 0, 0)(S_2) = \pi(1, 0, 0)(S_2) = \pi(0, 0,1)(S_2) = \pi(1, 0, 1)(S_2) = F$

$\pi(0, 0, 1)(S_3) = \pi(0, 1, 1)(S_3) = \pi(1, 0, 1)(S_3) = \pi(1, 1, 1)(S_3) = V$

$\pi(0, 0, 0)(S_3) = \pi(0, 1, 0)(S_3) = \pi(1, 0, 0)(S_3) = \pi(1, 1, 0)(S_3) = F$

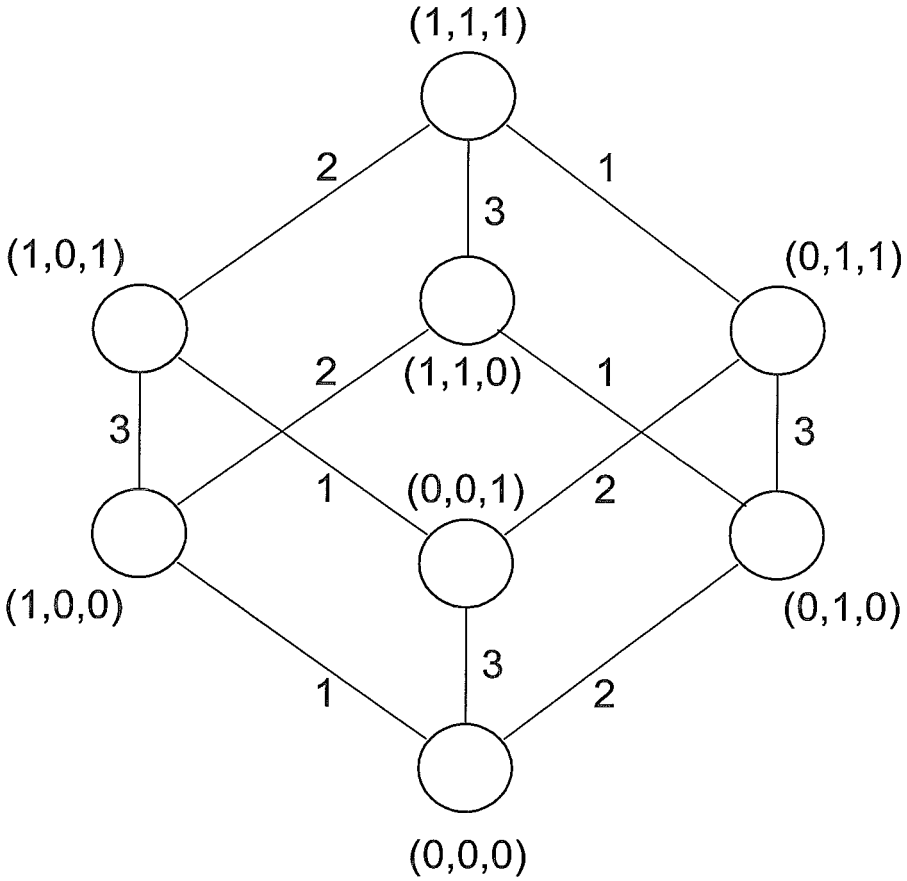

Figura 4.2 – Digrama dos mundos possíveis para o problema das crianças com lama na testa com n=3.

Na Figura 4.2 os vértices do grafo representam os mundos possíveis de Kripke e as arestas representam as relações R_i, onde i corresponde ao rótulo de identificação de cada aresta. Os ciclos que representariam a reflexividade das relações em todos os mundos, e as setas bidirecionais em cada aresta que representariam a simetria das relações foram omitidas, por simplificação.

4.2.2.2 Evolução do sistema

Quando o pai chega e diz que pelo menos uma criança está com a testa suja, as crianças deixam de acreditar no estado (0, 0, 0) e esse estado pode ser removido do

grafo junto com suas arestas, conforme mostrado na Figura 4.3. Dessa forma, a cada pergunta com resposta unânime negativa que o pai faz, outros estados vão sendo um a um removidos do grafo; mais especificamente, após a k-ésima pergunta com resposta negativa, os estados com k "1s" são removidos, acompanhando fielmente o raciocínio de cada criança. O conhecimento comum é atingido quando todos os agentes consideram apenas o estado real possível, ou seja, só ele resta no grafo. Isso se dará após a resposta afirmativa de todas as crianças sujas, e só então as crianças limpas saberão que não estão sujas.

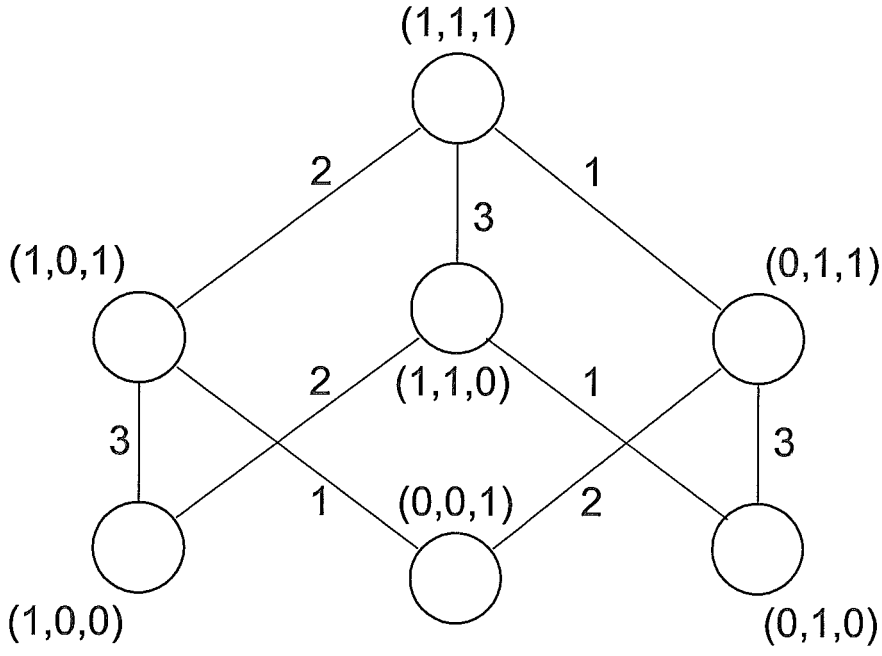

Figura 4.3 – Diagrama dos mundos possíveis após a eliminação do estado (0, 0, 0).

4.2.2.3 Propriedades do modelo síncrono

A afirmação feita anteriormente de que o pai das crianças não precisa ser considerado um agente no sistema pode ser explicada agora. Para o sistema, a afirmação inicial do pai ("pelo menos uma criança está com a testa suja") pode ser considerada como parte do conhecimento inicial de cada criança. Fora isso, o estado de conhecimento do pai sobre o sistema não afeta o raciocínio de nenhuma das crianças, e pode portanto ser desconsiderado.

As demais participações do pai no sistema correspondem às repetidas perguntas. Está claro que as perguntas modificam o estado de conhecimento das crianças, e não podem ser simplesmente eliminadas do sistema. O que acontece é que o tipo de informação que elas fornecem ao sistema é puramente temporal.

Para ilustrar essa idéia, convém pensar o que aconteceria se o problema fosse enunciado de tal forma que, ao invés de fazer repetidas perguntas, o pai simplesmente dissesse que a cada 10 minutos passados, se alguma criança já soubesse que estava suja deveria deixar a brincadeira e dirigir-se até ele, e fosse embora (supondo ainda que as crianças não trocam informações entre si sobre seus estados de conhecimento durante a ausência do pai, e que todas podem ver um único relógio). É fácil perceber que tudo se passaria da mesma forma, afinal a cada 10 minutos as crianças seriam capazes de concluir se alguma das outras já conseguiu descobrir sobre a sujeira em sua testa e seguir o mesmo raciocínio que seguiriam se o pai tivesse feito uma nova pergunta. Podemos concluir então que o sistema é síncrono, obedece a um relógio global e cada pulso desse relógio é simbolizado pelas perguntas feitas pelo do pai.

4.3 Representação para o Problema das Crianças com Lama na Testa utilizando LDC

Após o estudo detalhado do problema das crianças com lama na testa e de estruturas para representação do conhecimento, chegamos a uma representação para o este problema utilizando a lógica modal de ação LDC definida anteriormente. Veremos a seguir cada passo na construção da representação, e os motivos da adequação de LDC ao problema. Na seção 4.4 estão exemplos da representação proposta aplicada ao problema para **n**=3 crianças.

4.3.1 Descrição dos fatos e notação utilizada

As crianças são os agentes do nosso sistema. Para um problema de um grupo de **n** crianças onde **k** delas estão sujas, identificaremos sem perda de generalidade as crianças sujas por um número de 1 a **k**, e as demais crianças (limpas) por um número de **k+1** a **n**.

Vamos denotar por S_i o fato "a criança **i** está com a testa suja". Consequentemente $\neg S_i$ representará o fato "a criança i não está com a testa suja (está limpa)".

O fato anunciado pelo pai no início do sistema de que pelo menos uma das crianças está com a testa suja pode ser representado da seguinte forma na lógica

proposicional: $S_1 \vee S_2 \vee S_2 \vee ... \vee S_n$. Por simplificação, usaremos a seguinte notação: $\vee_{i \in G} S_i$.

Chamaremos de situação da criança i a sua situação em relação as condições de sua testa (limpa ou suja). Um agente tem necessariamente uma entre as duas situações.

4.3.2 Conhecimento inicial dos agentes

Nesta representação, o início do sistema se dá logo após o anuncio do pai de que há pelo menos uma criança suja. Todo o conhecimento e crenças que as crianças adquirem até o instante que sucede o anúncio do pai é considerado como o conhecimento inicial dos agentes do sistema. Nesse ponto, cada agente sabe:

1. Quais dentre as outras crianças estão sujas, visto que ele pode ver a testa de todos, menos na sua:
   - Para um agente i sujo ($i \leq \mathbf{k}$):

     $K_i S_j$, para $j \leq \mathbf{k}$, $i \neq j$; $K_i \neg S_j$, para $j > \mathbf{k}$.

   - Para um agente i limpo ($i > \mathbf{k}$):

     $K_i S_j$, para $j \leq \mathbf{k}$; $K_i \neg S_j$, para $j > \mathbf{k}$, $i \neq j$.
2. Que é de conhecimento comum no grupo que todo agente sabe a situação dos outros agentes, visto que qualquer agente pode ver todos os outros:

   $$C_G(S_i \rightarrow K_j S_i), \text{ para } i \neq j \text{ e } i, j=1, 2, ..., n;$$

   $$C_G(\neg S_i \rightarrow K_j \neg S_i), \text{ para } i \neq j \text{ e } i, j=1, 2, ..., n;$$
3. Que o fato anunciado pelo pai é de conhecimento comum no grupo de agentes:

   $$C_G(\vee_{i \in G} S_i)$$

4.3.3 Axiomas gerais do sistema

Apresentaremos a seguir os axiomas que restringirão os modelos para o sistema distribuído representado, impondo restrições sobre a aplicação modalidades de ação [tick] e [$send_i$], para todo $i \in G$. Tais restrições dependem da natureza da aplicação que está sendo representada, que neste caso corresponde ao problema das crianças com lama na testa.

De acordo com a descrição do problema, quando o pai faz a **k**-ésima pergunta, após **k-1** perguntas com resposta negativa, as **k** crianças sujas chegam simultaneamente à conclusão de que estão com a testa suja, e respondem "Sim". Como a resposta é anunciada a todo o grupo, ela se torna de conhecimento comum no grupo. Em decorrência dessa resposta afirmativa, as crianças limpas obtêm o conhecimento de que não estão sujas, e o sistema se encerra neste pulso do relógio global. Note que não importa de qual agente partiu primeiramente o anúncio, pois basta que uma das crianças sujas se pronuncie para que as crianças limpas possam concluir que estão limpas, visto que não importa a elas o número de crianças que se declaram sujas (fato que ela pode ver), mas sim o estado epistêmico dela no pulso no relógio global em que isso acontece;

1. Quando um agente chega à conclusão de que sua testa está suja, então ele imediatamente anuncia este fato ao grupo. Quando ocorre o anúncio do fato, todos os agentes do sistema alcançam o conhecimento comum do fato, e o relógio global pára (por esse motivo incluímos a fórmula $[tick]\bot$ no conjunto de fórmulas válidas após a ocorrência de K_iS_i, para i = 1, 2, ..., n). Essas duas propriedades do sistema decorrem diretamente da maneira como o sistema evolui (descrita no Capítulo 4, Seção 4.4.2).

   $\underline{G1}.\ K_iS_i \rightarrow (\ [send_i]C_G(K_iS_i)\ ) \wedge (\ [tick]\bot\ )$, para i = 1, 2, ..., n;

2. Quando um agente descobre que outro agente já concluiu que está sujo, se o agente em questão ainda não souber se está limpo ou sujo, então esse agente pode concluir que está limpo.

   $\underline{G2}.\ (\ K_iK_jS_j \wedge \neg K_iS_i\ ) \rightarrow (\ K_i\neg S_i\ )$, para $i \neq j$ e i, j=1, 2, ..., n;

3. São necessários também os axiomas de *frames* relativos ao conhecimento. Se um agente sabe um fato, esse fato continua sendo conhecido pelo agente após a execução de qualquer ação:

   $\underline{F1}.\ K_i\varphi \rightarrow [sendj]K_i\varphi$, para i, j = 1, 2, ..., n;

   $\underline{F2}.\ K_i\varphi \rightarrow [tick]K_i\varphi$, para i = 1, 2, ..., n;

4. O mesmo vale para um fato de conhecimento comum no grupo: se um fato é de conhecimento comum, ele o continua sendo após a execução de qualquer ação:

   $\underline{F3}.\ C_G\varphi \rightarrow [send_i]\ C_G\varphi$, para i = 1, 2, ..., n;

   $\underline{F4}.\ C_G\varphi \rightarrow [tick]\ C_G\varphi$;

4.3.4 Evolução do sistema (axiomas para [tick])

No estado inicial do sistema todos os agentes têm o "conhecimento inicial" definido na seção 6.1.2, que corresponde no enunciado do problema ao instante que sucede o anúncio do pai de que há pelo menos uma criança suja.

A partir de então, cada agente montará uma linha de raciocínio que evoluirá no sistema a cada pulso do relógio global, conforme apresentado no Capítulo 4, Seção 4.2.1. Considere um agente I do sistema (limpo ou sujo). Seja k_I o número de crianças sujas que I vê (que será k ou k-1, conforme a situação de I). Nos estados epistêmicos do agente I, vamos numerar essas k_I crianças de 1 a k_I para identificá-las (essa identificação das crianças vale somente nos estados epistêmicos de I, ou seja, cada agente identifica as crianças que vê sujas segundo sua própria ordenação). O agente I no estado inicial monta a seguinte linha de raciocínio:

$$B_I\,(\neg S_I \wedge B_1(\neg S_1 \wedge B_2(\neg S_2 \wedge \ldots B_{kI-2}\,(\neg S_{kI-2} \wedge B_{kI-1}\,(\neg S_{kI-1} \wedge K_{kI}S_{kI})\,)\ldots)\,)\,)\,)$$

Esse corresponde ao raciocínio ilustrado para **k**=3 no Capítulo 4, Seção 4.2.1. Acompanhando o raciocínio de I, teríamos: "Suponha que eu esteja limpo. Nesse caso, há k_I crianças sujas. A criança 1 pode imaginar (assim como eu) um estado em que ela esteja limpa, e nesse caso ela veria k_I-1 crianças sujas. Nesse estado epistêmico de 1, a criança 2 vê k_I-2 crianças sujas, e imaginando que todas as crianças seguem sucessivamente esse raciocínio, a criança k_I-1 imaginaria que a criança k_I não vê nenhuma criança suja, e portanto saberia que está suja (caso k=1)".

Quando ocorre uma ação [tick], o estado de conhecimeto de cada agente muda, da seguinte forma (axioma G3):

$$B_I\,(\neg S_I \wedge B_1(\neg S_1 \wedge B_2(\neg S_2 \wedge \ldots B_{kI-2}\,(\neg S_{kI-2} \wedge B_{kI-1}\,(\neg S_{kI-1} \wedge K_{kI}S_{kI})\,)\ldots)\,)\,)\,) \rightarrow$$

$$[\text{tick}]B_I\,(\neg S_I \wedge B_1(\neg S_1 \wedge B_2(\neg S_2 \wedge \ldots B_{kI-2}\,(\neg S_{kI-2} \wedge K_{kI-1}(S_{kI-1} \wedge K_{kI}S_{kI})\,)\ldots)\,)\,)\text{, para}$$

$$I=1\ ,2\ ,\ldots\ ,n.$$

Essa mudança corresponde ao conhecimento obtido quando todos respondem negativamente a uma pergunta feita pelo pai: significa que há pelo menos uma suja criança a mais do que o mínimo considerado na crença de uma criança k_I existente no estado epistêmico de cada criança I do sistema. Logo, toda criança I deixa de acreditar nesse estado, e passa a considerar o caso do estado epistêmico da criança k_I-1. Como visto na seção 4.4.2 a cada [tick] os mundos possíveis vão sendo gradativamente eliminados (após a **k**-ésima pergunta, os mundos com menos de **k** crianças sujas são eliminados).

4.4 Exemplos de representação do problema das crianças com lama na testa através de LDC para n=3 crianças

Veremos agora a representação do problema em um grupo de três crianças utilizando LDC. Para simplificar a notação e em virtude do pequeno número de crianças, na utilização do axioma G3 usaremos a mesma identificação geral das crianças, e não uma local ao estado epistêmico de cada agente.

4.4.1 Apenas uma criança suja: k=1

Fórmulas válidas no estado inicial:

$S_1, \neg S_2, \neg S_3$

$K_1(\neg S_2 \wedge \neg S_3)$

$K_2(S_1 \wedge \neg S_3)$

$K_3(S_1 \wedge \neg S_2)$

$C_G(S_1 \vee S_2 \vee S_3)$

$B_2(\neg S_2 \wedge K_1 S_1)$

$B_3(\neg S_3 \wedge K_1 S_1)$.

No próprio estado inicial, a criança 1 pode concluir que está suja:

$$C_G(S_1 \vee S_2 \vee S_3) \wedge K_1(\neg S_2 \wedge \neg S_3) \Rightarrow K_1 S_1$$

Aplicando $[send_1]$ e o axioma G1 para a criança 1:

$$K_1 S_1 \rightarrow ([send_1] C_G(K_1 S_1)) \wedge ([tick] \bot);$$

Obtemos um mundo onde vale:

$S_1, \neg S_2, \neg S_3$

$K_1(\neg S_2 \wedge \neg S_3)$

$K_2(S_1 \wedge \neg S_3)$

$K_3(S_1 \wedge \neg S_2)$

$C_G(S_1 \vee S_2 \vee S_3)$

$C_G(K_1 S_1)$; obtido pela aplicação de G1

Podemos usar o axioma G2:

$$(K_2 K_1 S_1 \wedge \neg K_2 S_2) \rightarrow (K_2 \neg S_2), (K_3 K_1 S_1 \wedge \neg K_3 S_3) \rightarrow (K_3 \neg S_3)$$

E conseqüentemente obter:

$K_2\neg S_2, K_3\neg S_3$

4.4.2 Duas crianças sujas: k=2

Fórmulas válidas no estado inicial:

$S_1, S_2, \neg S_3$ $\quad$ $C_G(S_1 \vee S_2 \vee S_3)$

$K_1(S_2 \wedge \neg S_3)$ $\quad$ $B_1(\neg S_1 \wedge K_2 S_2)$

$K_2(S_1 \wedge \neg S_3)$ $\quad$ $B_2(\neg S_2 \wedge K_1 S_1)$

$K_3(S_1 \wedge S_2)$ $\quad$ $B_3(\neg S_3 \wedge B_2(\neg S_2 \wedge K_1 S_1))$

Nada mais pode ser concluído nesse estado.

Aplicando uma ação [tick] e o axioma G3 passamos ao seguinte mundo:

$S_1, S_2, \neg S_3$ $\quad$ $C_G(S_1 \vee S_2 \vee S_3)$

$K_1(S_2 \wedge \neg S_3)$ $\quad$ $K_1(S_1 \wedge K_2 S_2)$

$K_2(S_1 \wedge \neg S_3)$ $\quad$ $K_2(S_2 \wedge K_1 S_1)$

$K_3(S_1 \wedge S_2)$ $\quad$ $B_3(\neg S_3 \wedge K_2(S_2 \wedge K_1 S_1))$

As crianças 1 e 2 já podem concluir que estão sujas:

$$K_1(S_1 \wedge K_2 S_2) \Rightarrow K_1 S_1,\ K_2(S_2 \wedge K_1 S_1) \Rightarrow K_2 S_2$$

Aplicando $[send_1]$ e o axioma G1 para a criança 1, passamos ao mundo:

$S_1, S_2, \neg S_3$ $\quad$ $K_1(S_1 \wedge K_2 S_2)$

$K_1(S_2 \wedge \neg S_3)$ $\quad$ $K_2(S_2 \wedge K_1 S_1)$

$K_2(S_1 \wedge \neg S_3)$ $\quad$ $K_1 S_1, K_2 S_2$

$K_3(S_1 \wedge S_2)$ $\quad$ $C_G(K_1 S_1)$

$C_G(S_1 \vee S_2 \vee S_3)$

Usando o axioma G2:

$$(K_3 K_1 S_1 \wedge \neg K_3 S_3) \rightarrow (K_3 \neg S_3)$$

Obtemos:

$$K_3 \neg S_3$$

Aplicando [send$_2$] e o axioma G1 para a criança 2, passamos ao mundo:

$S_1, S_2, \neg S_3$	$K_2(S_2 \wedge K_1 S_1)$
$K_1(S_2 \wedge \neg S_3)$	$K_1 S_1$
$K_2(S_1 \wedge \neg S_3)$	$K_2 S_2$
$K_3(S_1 \wedge S_2)$	$K_3 \neg S_3$
$C_G(S_1 \vee S_2 \vee S_3)$	$C_G(K_1 S_1)$
$K_1(S_1 \wedge K_2 S_2)$	$C_G(K_2 S_2)$

4.4.3 Três crianças sujas: k=3

Fórmulas válidas no estado inicial:

S_1, S_2, S_3	$C_G(S_1 \vee S_2 \vee S_3)$
$K_1(S_2 \wedge S_3)$	$B_1(\neg S_1 \wedge B_3(\neg S_3 \wedge K_2 S_2))$
$K_2(S_1 \wedge S_3)$	$B_2(\neg S_2 \wedge B_3(\neg S_3 \wedge K_1 S_1))$
$K_3(S_1 \wedge S_2)$	$B_3(\neg S_3 \wedge B_2(\neg S_2 \wedge K_1 S_1))$

Nada mais pode ser concluído nesse estado.

Aplicando [tick] e G3 passamos ao seguinte mundo:

S_1, S_2, S_3	$C_G(S_1 \vee S_2 \vee S_3)$
$K_1(S_2 \wedge S_3)$	$B_1(\neg S_1 \wedge K_3(S_3 \wedge K_2 S_2))$
$K_2(S_1 \wedge S_3)$	$B_2(\neg S_2 \wedge K_3(S_3 \wedge K_1 S_1)$
$K_3(S_1 \wedge S_2)$	$B_3(\neg S_3 \wedge K_2(S_2 \wedge K_1 S_1))$

Nada mais pode ser concluído nesse estado.

Aplicando [tick] e G3 novamente:

S_1, S_2, S_3	$C_G(S_1 \vee S_2 \vee S_3)$
$K_1(S_2 \wedge S_3)$	$K_1(S_1 \wedge K_3(S_3 \wedge K_2 S_2))$
$K_2(S_1 \wedge S_3)$	$K_2(S_2 \wedge K_3(S_3 \wedge K_1 S_1)$
$K_3(S_1 \wedge S_2)$	$K_3(S_3 \wedge K_2(S_2 \wedge K_1 S_1))$

As crianças 1, 2 e 3 já podem concluir que estão sujas:

$K_1(S_1 \wedge K_3(S_3 \wedge K_2 S_2)) \Rightarrow K_1 S_1$, $K_2(S_2 \wedge K_3(S_3 \wedge K_1 S_1) \Rightarrow K_2 S_2$

$K_3(S_3 \wedge K_2(S_2 \wedge K_1S_1) \Rightarrow K_3S_3$

Aplicando [$send_1$] e G1:

S_1, S_2, S_3	$K_2(S_2 \wedge K_3(S_3 \wedge K_1S_1)$
$K_1(S_2 \wedge S_3), K_2(S_1 \wedge S_3), K_3(S_1 \wedge S_2)$	$K_3(S_3 \wedge K_2(S_2 \wedge K_1S_1))$
$C_G(S_1 \vee S_2 \vee S_3)$	$K_1S_1, K_2S_2, , K_3S_3$
$K_1(S_1 \wedge K_3(S_3 \wedge K_2S_2))$	$C_G(K_1S_1)$

Aplicando [$send_2$] e G1:

S_1, S_2, S_3	$K_1(S_1 \wedge K_3(S_3 \wedge K_2S_2))$
$K_1(S_2 \wedge S_3)$	$K_2(S_2 \wedge K_3(S_3 \wedge K_1S_1)$
$K_2(S_1 \wedge S_3)$	$K_3(S_3 \wedge K_2(S_2 \wedge K_1S_1))$
$K_3(S_1 \wedge S_2)$	$K_1S_1, K_2S_2, , K_3S_3$
$C_G(S_1 \vee S_2 \vee S_3)$	$C_G(K_1S_1), C_G(K_2S_2)$

Aplicando [$send_3$] e G1:

S_1, S_2, S_3	$K_1(S_1 \wedge K_3(S_3 \wedge K_2S_2))$
$K_1(S_2 \wedge S_3)$	$K_2(S_2 \wedge K_3(S_3 \wedge K_1S_1)$
$K_2(S_1 \wedge S_3)$	$K_3(S_3 \wedge K_2(S_2 \wedge K_1S_1))$
$K_3(S_1 \wedge S_2)$	$K_1S_1, K_2S_2, , K_3S_3$
$C_G(S_1 \vee S_2 \vee S_3)$	$C_G(K_1S_1), C_G(K_2S_2), C_G(K_3S_3)$

Através dos exemplos, pode ser observado que as ações em LDC constituem um mecanismo capaz de representar mudanças de estados de uma forma condizente com o modelo de tarefa reativa apresentado no capítulo 3, devido aos conceitos os conceitos de estados e ações introduzidos. Tal fato juntamente com a semântica de mundos possíveis para conhecimento faz de LDC um modelo suficiente para representar o conhecimento presente em toda a evolução do sistema distribuído síncrono.

O conceito de ações e eventos porém se mostrará muito mais relevante ao se tratar sistemas assíncronos, o que será feito no próximo capítulo. Vale ressaltar que para representar o conhecimento do grupo foi utilizado o operador de conhecimento comum

definido em HALPERN e MOSES (1990), que está amarrado a sistemas síncronos. Ao lidar com sistemas assíncronos será necessário incorporar operadores adequados para conhecimento de grupo, em substituição a C_G.

Capítulo 5

Lógica de Conhecimento e Eventos para sistemas assíncronos

No capítulo 3 vimos como representar estados globais em sistemas distribuídos assíncronos através dos cortes consistentes. Vimos também como estabelecer relações temporais parciais entre os eventos de uma execução, e ainda como estabelecer relações temporais de passado e futuro de um estado global, ou seja, ordenações parciais sobre os cortes consistentes.

Neste capítulo apresentaremos uma lógica modal de conhecimento e tempo para sistemas distribuídos assíncronos, com o objetivo de representar o conhecimento de um grupo de agentes em um sistema distribuído assíncrono evoluindo no tempo. Foi com esta mesma intenção que examinamos uma lógica de conhecimento e tempo para sistemas síncronos no capítulo 2, e também que propusemos uma lógica similar no capítulo 4 que já envolvia ações, estando assim mais próxima da noção de eventos do modelo assíncrono. Agora, buscamos um modelo baseado em eventos que consiga lidar com conhecimento e com as várias ordenações possíveis para o conjunto de eventos de uma execução de um algoritmo assíncrono, que ditarão a noção de tempo particular de cada agente e também do sistema como um todo.

Para dar semântica a esta nova linguagem, apresentamos um modelo de Kripke para sistemas distribuídos assíncronos baseado no modelo de eventos e cortes consistentes apresentado no capítulo 3, e estabelecemos relações temporais e relações de conhecimento e crença sobre o modelo. Para expressar conhecimento e crença, procedemos da mesma forma vista anteriormente no modelo síncrono: através da noção de mundos indistinguíveis. As noções de conhecimento de grupo porém devem ser redefinidas para o modelo assíncrono, visto que as definições apresentadas anteriormente eram calcadas nas condições de simultaneidade particulares do modelo síncrono. Para expressar tempo utilizaremos operadores distintos dos já mencionados

neste trabalho, justamente no intuito de comportar a noção de sucessão temporal mais fraca, característica do modelo assíncrono.

Ao final, apresentamos uma aplicação da lógica para modelar alguns exemplos de sistemas distribuídos.

5.1 Tempo em Sistemas Distribuídos Assíncronos

Uma premissa dos sistemas síncronos é que todos os agentes envolvidos têm acesso a um relógio global que dita o tempo do sistema, ou que as computações ocorrem em turnos bem definidos, onde todos os agentes sabem em que turno estão a cada turno, e onde um turno ocorre simultaneamente para todos os agentes. Em outras palavras, é implicitamente assumido que o tempo é de conhecimento comum, de forma que todos os agentes executam suas computações sincronamente. Para este tipo de sistema, o tempo é sempre linear, o que faz com que o passado e o futuro de um turno esteja completamente determinado.

A lógica de conhecimento e tempo para sistemas síncronos proposta no já citado trabalho de Lehmann e apresentada no capítulo 2 permitia tratar o tempo através de uma seqüência ordenada de pulsos, o que era representado com a utilização do operador temporal "O". Em alguns sistemas porém a premissa de um relógio global compartilhado não se aplica. Ao contrário, pode-se ter muito pouca informação acerca do tempo. É o caso por exemplo de sistemas onde os agentes são processos sobre os quais nada pode ser garantido acerca da velocidade relativa de execução, ou talvez sobre o tempo de entrega das mensagens. Ao nos depararmos com o modelo para sistemas assíncronos apresentado no capítulo 3 percebemos que um pulso não representa muita coisa sob as circunstâncias do assincronismo, por vários motivos.

O primeiro é que como não há um relógio global cada agente tem sua própria noção de tempo, que lida com incertezas sobre o que acontece aos outros agentes do sistema e não é diretamente refletido em seu estado local. O segundo é que mesmo a noção de tempo global do sistema, correspondente a visão temporal que um observador externo teria olhando para os estados locais de cada um dos agentes no decorrer da computação pode considerar no máximo ordenações parciais sobre o conjunto dos estados globais do sistema no decorrer do tempo.

Isso acontece porque a noção temporal de um sistema assíncrono está baseada na ordem de ocorrência dos eventos do sistema em uma execução, e como foi visto no capítulo 3, apenas ordens parciais podem ser estabelecidas sobre o conjunto de

eventos de uma execução de um algoritmo assíncrono. Para obter então um modelo temporal capaz de lidar com conhecimento em sistemas assíncronos, vamos propor uma linguagem onde toda a noção temporal será baseada nos eventos e em suas várias ordenações possíveis durante uma execução. Para tal, novos operadores temporais condizentes com a noção de eventos e ordenações temporais parciais de eventos são apresentados e definidos, e relações baseadas nas ordenações temporais parciais de eventos serão utilizadas para dar semântica aos novos operadores.

5.2 Conhecimento em Sistemas Distribuídos Assíncronos

Para estabelecer a formalização de conhecimento em sistemas distribuídos assíncronos em concordância com o modelo baseado em eventos apresentado no capítulo 3 vamos considerar os estados de conhecimento dos agentes em diversos pontos ou estados globais de uma computação distribuída. De acordo com este modelo e diferentemente do modelo síncrono, agora existem várias possibilidades de cortes consistentes para uma mesma execução, sendo que apenas alguns corresponderão a estados globais reais em uma execução - alguns serão apenas possibilidades que nunca existirão.

A informação que cada agente tem acerca do sistema é representada por seu estado local, e por isso o estado local de um agente, bem como sua visão passada ou sua história local podem ser utilizados para definir o conhecimento de um agente em um determinado estado global.

Para definir as interpretações epistêmicas, devemos ainda considerar o conhecimento dos agentes nos possíveis estados globais consistentes do sistema. Existem diferentes maneiras de definir o que é uma interpretação epistêmica ou de conhecimento para um agente num determinado estado global. Uma possibilidade é fazer do conhecimento do agente uma função de seu estado local. Contudo, é desejável que os agentes não esqueçam os fatos de seu conhecimento, e neste caso podemos dizer que o estado é caracterizado ou coincide com a visão passada, na qual a ordem em que os eventos ocorrem não importa. Outra possibilidade seria ainda considerar uma interpretação epistêmica onde o estado de conhecimento do agente fosse baseado na história local, que constitui a seqüência de eventos acontecidos para um agente. Neste caso, a ordem de ocorrência dos eventos influenciaria no estado de conhecimento do agente.

A principal motivação de sistemas distribuídos é a cooperação entre os processos ou agentes. Em termos de conhecimento, tal cooperação caracteriza-se por obter alguma forma de "conhecimento de grupo". Em sistemas de memória distribuída, toda forma de colaboração entre os agentes deve se dar através da comunicação entre eles. Conseqüentemente, nesta classe de sistemas a comunicação é vista como uma forma de transferir conhecimento no sentido de prover tal conhecimento de grupo.

É mostrado em HALPERN & MOSES (1984) que mesmo quando a comunicação é garantida não é possível alcançar conhecimento comum em sistemas assíncronos. O conhecimento comum, segundo a definição de HALPERN & MOSES (1984) só pode ser alcançado em sistemas síncronos, onde ações coordenadas simultâneas são possíveis. Por este motivo outros tipos de conhecimento comum alcançáveis foram definidos, tais como *ε-conhecimento comum* em HALPERN & MOSES (1984) e *conhecimento comum concorrente* em PANANGADEN & TAYLOR (1992).

Em especial o conhecimento comum concorrente é um tipo de acordo alcançável em sistemas assíncronos que permite executar *ações concorrentes*. Sua definição baseia-se em uma relação de causalidade entre eventos e possíveis estados globais consistentes com esta relação, o que faz com que um agente não possa distinguir se um corte consistente é de fato um estado global real do sistema. A noção de conhecimento comum passa a ser a de *todo mundo concorrentemente sabe que* é verdade se todos os agentes sabem que a sentença é verdadeira em algum estado global possível (ou corte consistente) indistinguível.

Para falar de conhecimento do grupo num primeiro nível de acordo, intuitivamente, PANANGADEN & TAYLOR definem que *todo mundo concorrentemente sabe que uma fórmula é verdadeira se todos os agentes sabem que ela é verdadeira em algum estado global possível (ou corte consistente) indistinguível.* Assim sendo, aplicações onde os processos precisam chegar a um acordo sobre uma propriedade de um estado global do sistema podem ser entendidas em termos de conhecimento comum concorrente.

Uma semântica formal para uma lógica de conhecimento comum concorrente foi definida em PANANGADEN & TAYLOR (1992), embora nenhum sistema axiomático tenha sido apresentado. A sintaxe seria a mesma definida em LEHMANN (1984) para o caso síncrono, porém com a adição de operadores modais para conhecimento comum concorrente.

Decidimos incluir na Lógica de Conhecimento e Eventos apenas os operadores de conhecimento de um agente K_i e B_i. Para a representação de conhecimento do grupo, recomenda-se o uso dos operadores para conhecimento de grupo apresentados em PANANGADEN & TAYLOR (1992), porém quaisquer operadores consistentes com o modelo baseado em eventos para sistemas distribuídos assíncronos poderia ser adequadamente incorporado para o tratamento de conhecimento de grupo na Lógica de Conhecimento e Eventos.

5.3 Apresentação da Lógica de Conhecimento e Eventos em Sistemas Distribuídos Assíncronos

O tratamento da noção de conhecimento será o mesmo utilizado em LEHMANN (1984), anteriormente apresentado no capítulo 2: o dos estados indistinguíveis. Para o tratamento da noção temporal de cada agente e do sistema será proposto um modelo baseado em eventos, de acordo com o modelo para sistemas distribuídos assíncronos proposto em LAMPORT (1978) e apresentado no capítulo 3.

A linguagem utilizada é a da lógica proposicional modal para um número m de agentes (L_m) apresentada no Capítulo 2 e acrescida das modalidades $\Box$, $\Diamond$, $[v_i]$, $\langle v_i \rangle$, (v_i), *UNTIL*, $\Box_i$, $\Diamond_i$, onde i=1, 2,..., m.

5.3.1 Símbolos

Os símbolos da linguagem são:

1. Um conjunto Φ enumerável de símbolos proposicionais;
2. Pontuação: "(" e ")";
3. Conectivos: "$\neg$", "$\wedge$", "$\vee$" e "$\rightarrow$";
4. Modais de conhecimento e crença: K_i, B_i i = 1, 2, 3, ..., m (um para cada agente);
5. Modais temporais $\Box_i$, $\Diamond_i$, $[v_i]$, $\langle v_i \rangle$, (v_i), i = 1, 2, ..., m (um para cada agente);
6. Modais temporais $\Box$, $\Diamond$, *UNTIL*;

5.3.2 Operadores

Os operadores da linguagem são definidos da seguinte forma:

1. Os conectivos seguem as definições da lógica proposicional;

2. O operadores modais "K_i" e "B_i" seguem as definições da lógica de conhecimento L_m;
3. O operador modal "$[v_i]$" representa a validade após a ocorrência do evento v_i, ou seja: $[v_i]\varphi$ indica que φ sempre vale a partir da ocorrência do evento v_i, para i=1, 2, 3, ..., m;
4. O operador modal "$<v_i>$" representa a validade eventual após a ocorrência do evento v_i, ou seja: $<v_i>\varphi$ indica que φ eventualmente vale após a ocorrência do evento v_i, para i=1, 2, 3, ..., m. Em algum momento após a ocorrência do evento v_i, φ sempre será valido;
5. O operador modal "(v_i)" representa a validade imediata após a ocorrência do evento v_i, ou seja: $(v_i)\varphi$ indica que φ vale no estado decorrente da ocorrência do evento v_i que sucede o estado atual, para i=1, 2, 3, ..., m (o que não impede que φ torne-se falso posteriormente);
6. O operador modal "$\Box$" representa a validade após o estado atual, ou seja: validade em qualquer estado sucessor do estado global atual. Em outras palavras, $\Box\varphi$ indica que φ vale após o primeiro evento que modifique o estado global (φ pode tornar-se falso posteriormente);
7. O operador modal "$\Diamond$" representa possibilidade após o estado global, ou seja: validade em algum estado sucessor do estado global atual. Em outras palavras, $\Diamond\varphi$ indica que existe algum evento possível de ocorrer após o qual φ vale (φ pode tornar-se falso posteriormente);
8. O operador modal "*UNTIL*" representa a relação de validade condicionada entre uma fórmula e um evento, da seguinte forma: φ *UNTIL* v_i indica que φ vale até que o evento v_i aconteça.
9. O operador modal "$\Box_i$" representa a validade após a ocorrência do primeiro evento para o agente i, ou seja: $\Box_i\varphi$ indica que φ vale no estado acessível pelo primeiro evento que acontecer para o agente i, onde i=1, 2, ..., m (φ pode tornar-se falso posteriormente);
10. O operador modal "$\Diamond_i$" representa possibilidade após a ocorrência do primeiro evento para o agente i, ou seja: $\Diamond_i\varphi$ indica que φ possivelmente vale no estado acessível pelo primeiro evento que acontecer para o agente i, onde i=1, 2, ..., m (φ pode tornar-se falso posteriormente);

5.3.3 Fórmulas

As fórmulas da linguagem são descritas pelas seguintes regras:

1. Todo símbolo proposicional de Φ é uma fórmula, chamada fórmula atômica.
2. Se φ é uma fórmula, então $(\neg\varphi)$ também é uma fórmula.
3. Se φ e ψ são fórmulas, então $(\varphi \wedge \psi)$, $(\varphi \vee \psi)$ e $(\varphi \rightarrow \psi)$ também são fórmulas.
4. Se φ é uma fórmula, então $K_i\varphi$ e $B_i\varphi$ também são fórmulas, para i=1, 2, 3, ..., m.
5. Se φ e ψ são fórmulas e v_i é um evento, então $[v_i]\varphi$, $<v_i>\varphi$, $(v_i)\varphi$, φ *UNTIL* v_i, $\Box\varphi$, $\Diamond\varphi$, $\Box_i\varphi$, $\Diamond_i\varphi$ também são fórmulas, para i=1, 2, ..., m.
6. Nada é uma fórmula, a não ser que seja forçado por um dos itens acima.

5.4 Semântica para Lógica de Conhecimento e Eventos em Sistemas Distribuídos Assíncronos

Para dar uma interpretação de Kripke às modalidades de conhecimento e eventos necessitamos estabelecer um conjunto apropriado de mundos possíveis e de relações entre os mundos que suporte as modalidades apresentadas para o modelo assíncrono. Para expressar conhecimento e crença, procedemos da mesma forma que no modelo síncrono: preservamos a noção de mundos possíveis, utilizando relação de indistinguibilidade. Para expressar tempo estabelecemos relações temporais parciais sobre o modelo, afinal sua natureza impede que o modelo comporte relações temporais mais fortes.

Um sistema distribuído segundo a especificação apresentada no capítulo 3 pode ser considerado um *Frame* para uma estrutura Kripke onde:

- Os mundos possíveis são os cortes consistentes ou estados globais do sistema;
- Os fatos básicos do sistema são as primitivas;
- As relações entre os estados globais são descritas nas subseções seguintes.

5.4.1 Relações entre estados globais de um sistema distribuído assíncrono

5.4.1.1 Relação de indistinguibilidade baseada na visão passada ($\sim_i$)

Em um sistema assíncrono, os mundos possíveis são os cortes consistentes de um conjunto de execuções assíncronas. Vamos definir a relação de indistinguibilidade entre os cortes consistentes de uma computação distribuída utilizando o conceito de "estados indistinguíveis" apresentado anteriormente e revisto na definição da relação de indistinguibilidade no Capítulo 2.

Dois estados globais ou cortes consistentes s e s' são indistinguíveis em relação ao agente n_i ($s \sim_i s'$) se n_i tem a mesma visão passada em ambos os cortes s e s'. A relação para dois estados de uma execução de PIF está ilustrada na figura 5.1.

Como conseqüência própria da definição, a relação de indistinguibilidade é reflexiva, transitiva e simétrica.

Basear a relação de indistinguibilidade na visão passada implica que o agente nunca esquece os fatos que já sabe, mas a ordem em que ele tomou conhecimento dos fatos não importa. O mais relevante é que o argumento intuitivo inicial seja preservado: a relação $\sim_i$ relaciona estados globais onde o conhecimento do agente i não permite que ele consiga distinguir entre os estados relacionados.

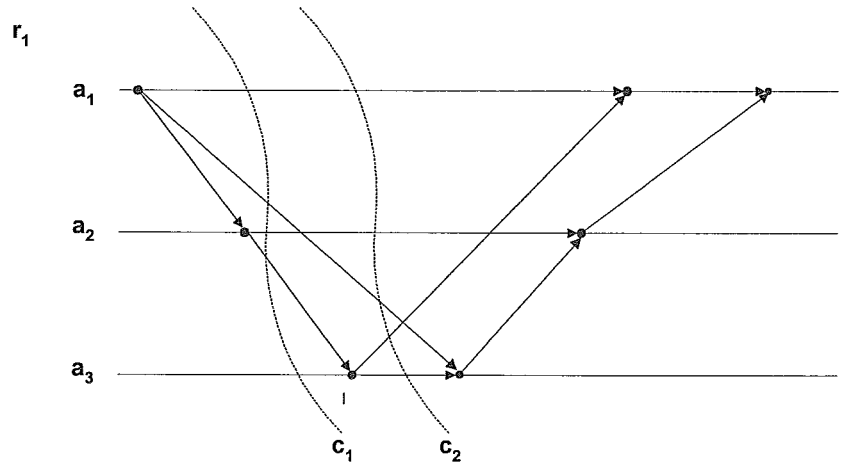

Figura 5.1: Os estados globais E_1 e E_2 determinados pelos cortes c_1 e c_2 diferem pela ocorrência do evento v para o agente a_3. Como a_1 e a_2 não percebem que o evento ocorreu e que o estado global mudou, para eles estes dois estados do sistema são indistinguíveis:

$E_1 \sim_1 E_2$ e $E_1 \sim_2 E_2$.

5.4.1.2 Relação de acessabilidade por eventos (R_v^i)

Dizemos que um estado global *s* é *acessível* a partir de algum outro estado global *t* através do evento *v* se estando no estado *s* a ocorrência do evento *v* transforma o estado global de *s* para *t*. Ou seja: podemos obter *t* a partir de *s* pela ocorrência de *v*. Diz-se nesse caso que *t* é *acessível* a partir de *s* através do evento *v*.

Baseado no conceito apresentado de estados acessíveis, definiremos agora a relação R_v^i entre estados globais de um sistema distribuído: "Dois estados globais relacionam-se através da Relação de acessibilidade por eventos (R_v^i) se e somente se tais estados diferem apenas pela ocorrência de um evento para o agente i". R_v^i é uma relação temporal, visto que eventos acontecem em um determinado tempo e ordenar estados globais de acordo com o acontecimento de um evento é estabelecer qual estado antecede outro na computação. Essa ordenação porém será sempre parcial, dadas as propriedades do modelo assíncrono que inviabilizam o estabelecimento de ordenações temporais totais sobre os eventos de uma execução.

Para formalizar a definição de R_v^i, será utilizada a primeira definição de estado global vista no capítulo 3:

Dois estados globais s e s' relacionam-se através de R_v^i em relação a um evento v_i (ocorrido para o agente n_i) se e somente se existe uma ordem total $<$ consistente com $\prec^+$ tal que:

- Ou s é um estado inicial e Passado(v_i) é vazio (v_i é o primeiro evento ocorrido para n_i) e s'= estado_sistema (v_i, v_2) para algum v_2 segundo a ordenação <);
- Ou s' é estado final e futuro(v_i) é vazio (v_i é o último evento ocorrido para n_i) e s = estado_sistema (v_1, v_i) (para algum v_1 segundo a ordenação <);
- Ou s= estado_sistema (v_1, v_i) e s'= estado_sistema (v_i, v_2) segundo a mesma ordenação total, ou seja: v_1, v', v_2 são eventos consecutivos nesta ordem segundo <.

Segundo esta definição, s R_v^i s' implica que s é o estado imediatamente antes do evento v_i acontecer e s' é o estado imediatamente depois de v_i acontecer. Nesse caso, todos os agentes n_k da computação distribuída tais que $n_k \neq n_i$ permanecem em s' no mesmo estado local em que se encontravam em s. O estado das arestas que não partem de n_i (Φ_{kj} onde $k \neq i$) também permanece em s' como estava em s. O estado das arestas Φ_{ij} que fazem a comunicação de n para seus vizinhos porém pode ser modificado em s' devido ao envio de mensagens ocasionado pelo evento v_i. O estado local de n_i é modificado devido a ocorrência de v_i sendo portanto diferente em s e s'. A figura 5.2 mostra dois estados da mesma execução de PIF vista na figura 5.1 que se relacionam através de R_v^i.

A relação de acessibilidade por eventos R_v^i tem propriedades temporais, visto que um estado *t* ser acessível de um estado *s* através do evento *v* implica que *t* sucede *s* na computação distribuída. Por conta de estabelecer uma sucessão direta temporal entre os estados, a relação R_v^i é não-reflexiva (não permite estacionar no tempo) e anti-simétrica (não permite voltar no tempo). E como, também por definição, R_v^i relaciona estados que representam sucessão imediata no tempo, também não será uma relação transitiva.

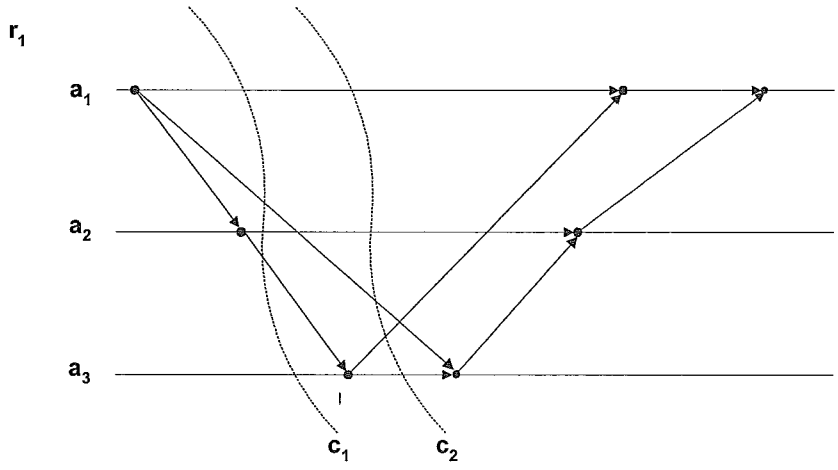

Figura 5.2: Diferentemente de a_1 e a_2, a_3 percebe a ocorrência do evento v. Os estados globais E_1 e E_2 determinados pelos cortes c_1 e c_2 diferem pela ocorrência de v para a_3, logo: $E_1 R_v^3 E_2$.

É interessante reparar no caráter "complementar" das relações de indistinguibilidade e acessabilidade por eventos. Enquanto $\sim_i$ relaciona estados globais nos quais o estado local do agente n_i é o mesmo, R_v^i só relacionará dois estados em que especificamente o estado local do agente i será diferente, visto que a ocorrência de um evento para o agente i modifica apenas seu próprio estado local. Isso implica que estados que se relacionam através de $\sim_i$ não se relacionam através de R_v^i, o que pode ser percebido também na figura 5.2.

Sejam s e s' dois estados globais consistentes. Podemos estabelecer as seguintes premissas sobre os estados relacionados por $\sim_i$ e R_v^i:

1. $s \sim_i s' \rightarrow \neg\, s\, R_v^i\, s'$
2. $s\, R_v^i\, s' \rightarrow \neg\, (s\, R_v^j\, s')$ para todo $j \neq i$
3. $s\, R_v^i\, s' \rightarrow s \sim_j s'$ para todo $j \neq i$

A afirmativa 1 é conseqüência da discussão anterior, e as premissas dois e três decorrem do fato que um estado s' só pode ser acessível por outro estado anterior s através da ocorrência de um e apenas um evento, e um evento ocorre para um único agente, por definição.

Dito isso, vamos agora definir outras relações a partir de R_v^i e ~i, tirando proveito de suas propriedades complementares e potencializando as propriedades temporais de R_v^i.

5.4.1.3 Relação temporal local baseada em eventos R_+^i

Para um agente i, construiremos a relação R_+^i definida a partir das relações R_v^i e $\sim_i$. Uma vez que $s\, R_v^i\, s' \leftrightarrow \neg\, (\, s \sim_i s'\,)$ para um mesmo agente i, podemos obter

uma relação local ao agente i dada por $R_+^i = [\ \sim_i \circ (\cup R_v^i) \circ \sim_i\ ]^+$,para um determinado i correspondente a um agente do grupo. R_+^i representa o fecho da composição da relação de indistinguibilidade com a união de todas as relações de acessabilidade por eventos do agente i composto novamente com a relação de indistinguibilidade para o mesmo agente. O propósito de R_+^i é relacionar temporalmente todos os estados globais do ponto de vista de um determinado agente. Os estados indistinguíveis porém não podem ser totalmente ordenados no tempo, logo a relação retratará uma ordenação parcial. A relação será exemplificada a seguir.

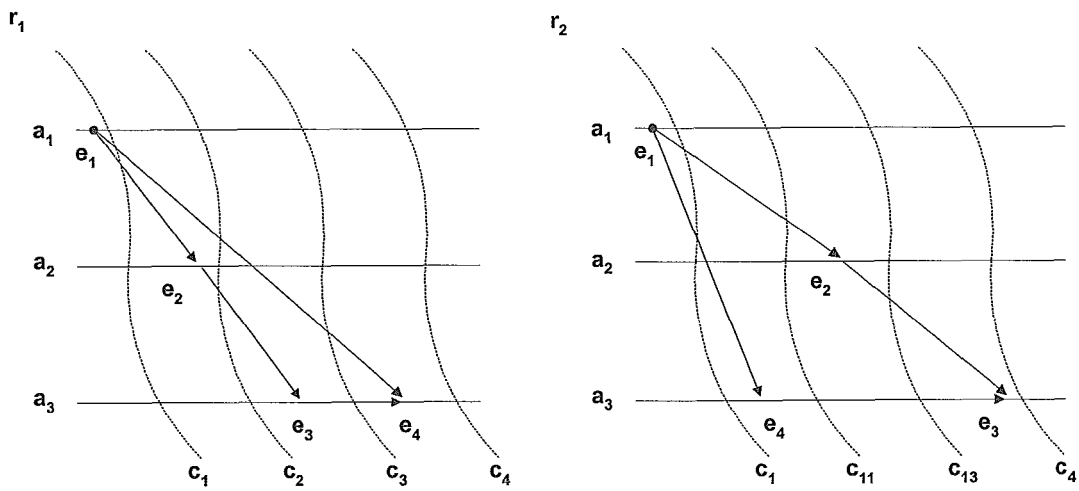

Figura 5.3: Grafo de precedência para os eventos iniciais de duas execuções distintas de PIF dentre as já apresentadas no capítulo 3. O estado inicial correspondente ao corte consistente c_0 não está representado.

Vamos descrever as relações $\sim_i$, R_v^i e R_+^i para as duas execuções parciais de PIF apresentadas na figura 5.3:

Execução r_1:

Para o agente a_1:

$\sim_1 =$ $\{(c_0, c_0), (c_1, c_1), (c_2, c_2), (c_3, c_3), (c_4, c_4), (c_1, c_2), (c_1, c_3), (c_1, c_4), (c_2, c_1),$
$(c_2, c_3), (c_2, c_4), (c_3, c_1), (c_3, c_2), (c_3, c_4), (c_4, c_1), (c_4, c_2), (c_4, c_3)\}$

$R_{e1}^{\ 1} =$ $\{(c_0, c_1)\}$

$R_+^{\ 1} =$ $\{(c_0, c_1), (c_0, c_2), (c_0, c_3), (c_0, c_4)\}$

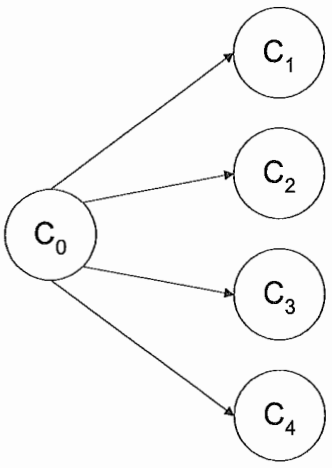

Figura 5.4: Grafo da Relação R_+^1 sobre os estados globais da execução r_1.

Para o agente a_2:

$\sim_2 =$ $\{(c_0, c_0), (c_1, c_1), (c_2, c_2), (c_3, c_3), (c_4, c_4), (c_0, c_1), (c_1, c_0), (c_2, c_3), (c_2, c_4), (c_3, c_2), (c_3, c_4), (c_4, c_2), (c_4, c_3)\}$

$R_{c2}^2 =$ $\{(c_1, c_2)\}$

$R_+^2 =$ $\{(c_1, c_2), (c_0, c_2), (c_1, c_3), (c_1, c_4), (c_0, c_3), (c_0, c_4)\}$

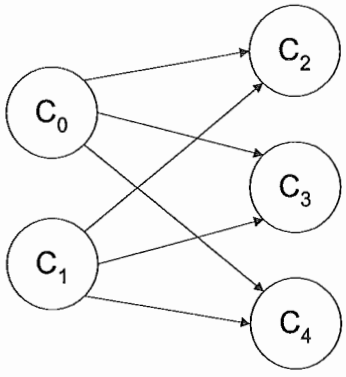

Figura 5.5: Grafo da Relação R_+^2 sobre os estados globais da execução r_1.

Para o agente a_3:

$\sim_3 =$ $\{(c_0, c_0), (c_1, c_1), (c_2, c_2), (c_3, c_3), (c_4, c_4), (c_0, c_1), (c_0, c_2), (c_1, c_0), (c_1, c_2), (c_2, c_0), (c_2, c_1)\}$

$R_{e3}^3 =$ $\{(c_2, c_3)\}$

$R_{e4}^3 =$ $\{(c_3, c_4)\}$

$R_+^3 =$ $\{(c_2, c_3), (c_0, c_3), (c_1, c_3), (c_3, c_4), (c_2, c_4), (c_0, c_4), (c_1, c_4)\}$

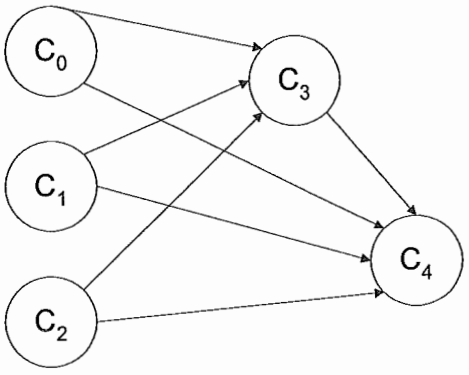

Figura 5.6: Grafo da Relação R_+^3 sobre os estados globais da execução r_1.

Execução r_2:

Para o agente a_1:

$\sim_1 =$ $\{(c_0, c_0), (c_1, c_1), (c_{11}, c_{11}), (c_{13}, c_{13}), (c_4, c_4), (c_1, c_{11}), (c_1, c_{13}), (c_1, c_4), (c_{11}, c_1), (c_{11}, c_{13}), (c_{11}, c_4), (c_{13}, c_1), (c_{13}, c_{11}), (c_{13}, c_4), (c_4, c_1), (c_4, c_{11}), (c_4, c_{13})\}$

$R_{e1}^1 =$ $\{(c_0, c_1)\}$

$R_+^1 =$ $\{(c_0, c_1), (c_0, c_{11}), (c_0, c_{13}), (c_0, c_4)\}$

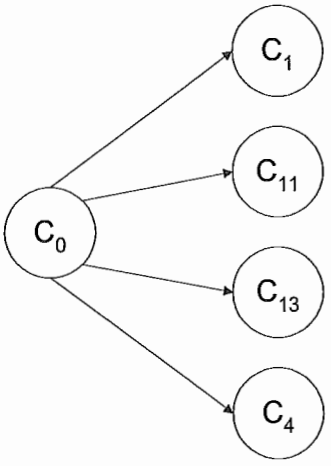

Figura 5.7: Grafo da Relação R_+^1 sobre os estados globais da execução r_2.

Para o agente a_2:

$\sim_2 =$ $\{(c_0, c_0), (c_1, c_1), (c_{11}, c_{11}), (c_{13}, c_{13}), (c_4, c_4), (c_0, c_1), (c_0, c_{11}), (c_1, c_0), (c_1, c_{11}), (c_{13}, c_4), (c_4, c_{13})\}$

$R_{e2}^2 =$ $\{(c_{11}, c_{13})\}$

$R_+^2 =$ $\{(c_{11}, c_{13}), (c_0, c_{13}), (c_1, c_{13}), (c_{11}, c_4), (c_0, c_4), (c_1, c_4)\}$

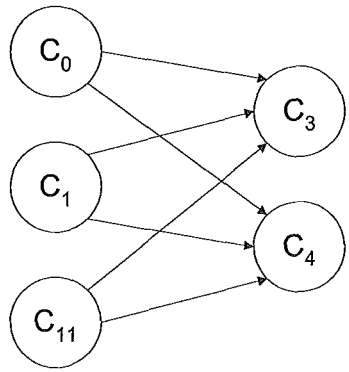

Figura 5.8: Grafo da Relação R_+^2 sobre os estados globais da execução r_2.

Para o agente a_3:

$\sim_3 =$ $\{(c_0, c_0), (c_1, c_1), (c_{11}, c_{11}), (c_{13}, c_{13}), (c_4, c_4), (c_0, c_1), (c_1, c_0), (c_{11}, c_{13}), (c_{13}, c_{11})\}$

$R_{e4}^3 =$ $\{(c_1, c_{11})\}$

$R_{e3}^3 =$ $\{(c_{13}, c_4)\}$

$R_+^3 =$ $\{(c_1, c_{11}), (c_0, c_{11}), (c_{13}, c_4), (c_1, c_{13}), (c_0, c_{13}), (c_{11}, c_4), (c_1, c_4), (c_0, c_4)\}$

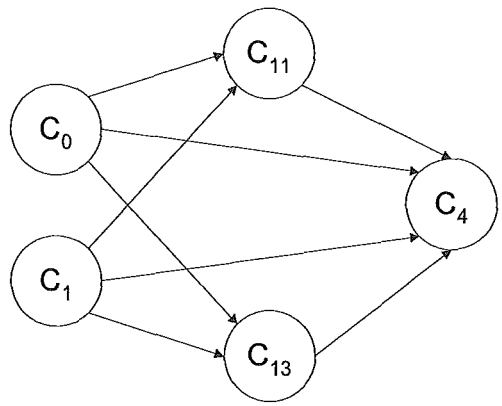

Figura 5.9: Grafo da Relação R_+^2 sobre os estados globais da execução r_2.

Para cada agente i R_+^i descreve como o agente percebe a execução, ou seja, como o estado local do agente evolui conforme os eventos ocorrem no tempo. Como posso ter várias possibilidades de cortes consistentes e conseqüentemente estados globais, a relação também reflete as várias possibilidades de sucessor para um estado. Para o agente i, a relação dá uma ordem de precedência temporal entre os estados, parcial devido a se ter várias possibilidades para o sucessor (e também para o antecessor) temporal de um estado.

Por envolver ordem temporal de eventos, temos que R_+^i é irreflexiva e antisimétrica. A transitividade de R_+^i é forçada pela definição.

5.4.1.4 Relação temporal global baseada em eventos $R_{\cup}$

Para representar a noção de tempo global, definiremos uma relação temporal global baseada em eventos como a união de R_v^i para todo evento v_i, para todo agente i do grupo. Chamaremos esta relação de $R_{\cup}$, onde $R_{\cup} = R_v^1 \cup R_v^2 \cup ... \cup R_v^n$.

Vamos descrever a relação $R_{\cup}$ para a execução de PIF da figura 5.4.

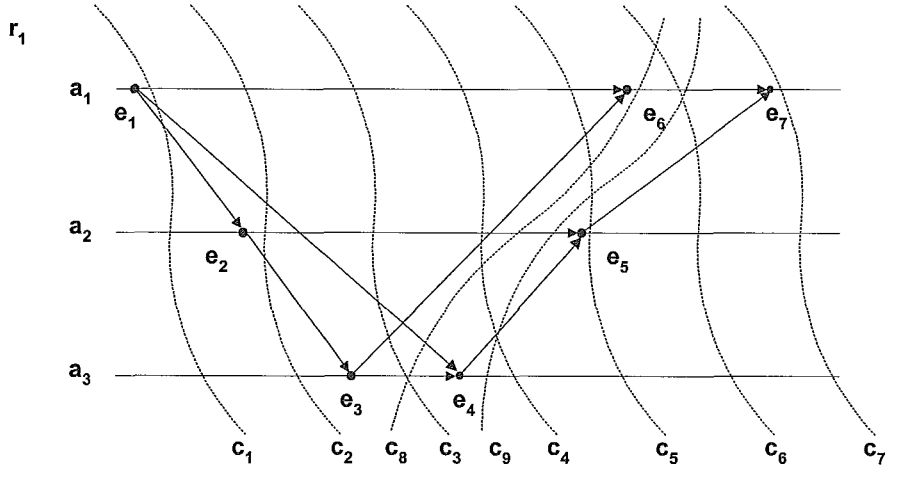

Figura 5.10: Grafo de precedência para uma das possíveis execuções de PIF em um sistema distribuído assíncrono com três agentes.

$R_{e1}^1 = \{(c_0, c_1)\}$ $R_{e6}^1 = \{(c_3, c_8), (c_4, c_9), (c_5, c_6)\}$ $R_{e7}^1 = \{(c_6, c_7)\}$

$R_{e2}^2 = \{(c_1, c_2)\}$ $R_{e5}^2 = \{(c_4, c_5), (c_9, c_6)\}$

$R_{e3}^3 = \{(c_2, c_3)\}$ $R_{e4}^3 = \{(c_3, c_4), (c_8, c_9)\}$

$R_{\cup} = \{(c_0, c_1), (c_1, c_2), (c_2, c_3), (c_3, c_4), (c_3, c_8), (c_4, c_5), (c_4, c_9), (c_8, c_9), (c_5, c_6), (c_9, c_6), (c_6, c_7)\}$

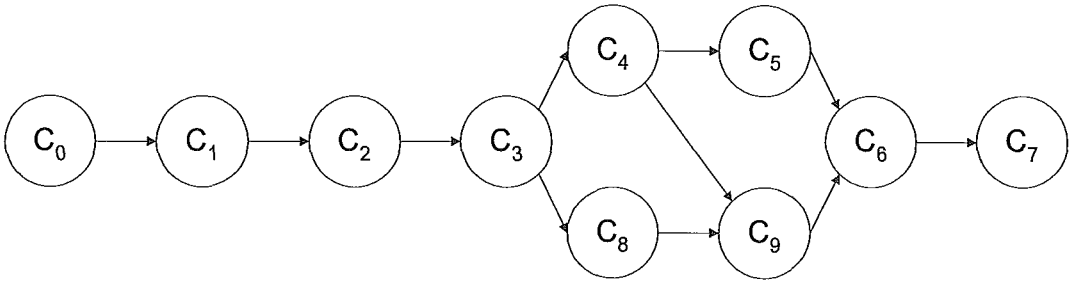

Figura 5.11: Grafo da Relação $R_{\cup}$ sobre os estados globais da execução de PIF apresentada na figura 5.10.

Para cada conjunto de cortes de uma execução, $R_{\cup}$ dá uma ordenação temporal global dos estados (e conseqüentemente dos eventos). Como não é possível definir a ordem exata dos eventos mesmo para uma única execução, devido às várias possibilidades de cortes consistentes, a relação $R_{\cup}$ reflete as possibilidades no sentido de que seu fecho transitivo não é linear, e sim ramificado. O estado c_3 por exemplo pode

ser sucedido tanto por c_4 quanto por c_8, o que é indicado pela presença dos pares (c_3, c_4) e (c_3, c_8) em $R_\cup$. O mesmo vale para o antecessor do estado c_6, que pode ser tanto c_5 quanto c_9.

Por representar a união das relações R_v^i, $R_\cup$ é irreflexiva, antisimétrica e não transitiva, o que pode ser visto também no exemplo ilustrado nas figuras 5.10 e 5.11.

5.4.2 Frame

Um *frame* $F = (S, \sim_i, R_v^i, R_+^i, R_\cup)$, i=1, 2, 3, ..., m é uma estrutura onde:

- S é um conjunto de *estados* ou *mundos possíveis*;
- $\sim i$, R_v^i, R_+^i, $R_\cup$ são relações binárias em S, ou seja, um conjunto de pares de elementos de S onde i=1, 2, 3, ..., m ($\sim_i \subseteq S \times S$, $R_v^i \subseteq S \times S$, $R_+^i \subseteq S \times S$, $R_\cup \subseteq S \times S$);

5.4.3 Modelo

Um modelo M sobre $F = (S, \sim_i, R_v^i, R_+^i, R_\cup)$ é um par $M = (F, \pi)$, onde π é uma interpretação (função de valoração) que associa valores verdade às primitivas de Φ em cada estado de S, isto é, $\pi: \Phi \times S \rightarrow \{V, F\}$.

5.4.4 Satisfatibilidade

Uma fórmula φ de é verdadeira em (M, s), ou seja, é verdadeira em um estado $s \in S$ para um modelo M quando:

1. $M, s \models p$ se e somente se $\pi(s, p) = V$, onde $p \in \Phi$;
2. $M, s \models \neg\varphi$ se e somente se não é o caso que $M, s \models \varphi$;
3. $M, s \models \varphi \wedge \psi$ se e somente se $M, s \models \varphi$ e $M, s \models \psi$;
4. $M, s \models K_i\varphi$ se e somente se para todo $t \in S$ tal que $(s, t) \in \sim_i$ temos que $M, t \models \varphi$;
5. $M, s \models B_i\varphi$ se e somente se existe t tal que $t \in S$ e $(s, t) \in \sim_i$, para o qual temos que $M, t \models \varphi$;
6. $M, s \models [v_i]\varphi$ se e somente para todos os estados que sucedem o acontecimento de v_i vale φ, ou seja: para todo par de estados s', s'' tal que

s' R_v^i s'' onde s' $\in$ Futuro(s) M, s'' $\models \varphi$, e para todo estado s''' tal que s'' R_+^i s''' M, s''' $\models \varphi$

7. M, s $\models \langle v_i \rangle \varphi$ se e somente existe algum estado que sucede o acontecimento de v_i a partir do qual φ sempre vale, ou seja, existe um par de estados s', s'' tal que s' R_v^i s'' onde s' $\in$ Futuro(s) e: ou M, s'' $\models \varphi$ e para todo s''' onde s'' R_+^i s''' vale M, s''' $\models \varphi$, ou existe um estado s''' tal que s'' R_+^i s''' e M, s''' $\models \varphi$, e para todo s'''' tal que s''' R_+^i s'''' M, s'''' $\models \varphi$
8. M, s $\models (v_i)\varphi$ se e somente se existe s' tal que s R_v^i s' e M, s' $\models \varphi$
9. M, s $\models \Box\varphi$ se e somente se para todo s' tal que s $R_\cup$ s' e M, s' $\models \varphi$
10. M, s $\models \Diamond\varphi$ se e somente se existe s' tal que s $R_\cup$ s' e M, s' $\models \varphi$
11. M, s $\models \varphi$ *UNTIL* v_i se e somente se para todos os estados que sucedem s e antecedem o acontecimento de v_i vale φ, ou seja: para todo par de estados s', s'' tal que s' R_v^i s'' onde s' $\in$ Futuro(s), para todo estado s''' tal que s''' $\in$ Futuro(s) e s''' $\in$ Passado(s'), M, s''' $\models \varphi$.
12. M, s $\models \Box_i \varphi$ se e somente se para todo par de estados s' e s" tal que s' R_v^i s" onde s $\sim_i$ s', M, s" $\models \varphi$.
13. M, s $\models \Diamond_i \varphi$ se e somente se existe um par de estados s' e s" tal que s' R_v^i s", onde s $\sim_i$ s' e M, s" $\models \varphi$.

Seja M = (F, π) um modelo para F. Dizemos que M satisfaz φ se existe algum mundo s $\in$ S tal que M, s $\models \varphi$. Dizemos que φ é satisfatível se existe algum modelo que o satisfaça, caso contrário, dizemos que φ é insatisfatível. Uma fórmula φ é válida em um Modelo M se φ é satisfeita em todos os estados de M (para todo s $\in$ S, M, s $\models \varphi$). Uma fórmula φ é válida em F se φ é válida em todos os modelos sobre F (para todos M e s, M, s $\models \varphi$).

5.4.5 Fórmulas válidas

Apresentaremos agora algumas fórmulas válidas na linguagem que acabamos de definir, bem como algumas regras de inferência. As fórmulas estão agrupadas em quatro categorias.

5.4.5.1 Distribuição dos operadores modais sobre a implicação

As fórmulas 1 a 4 indicam que qualquer um dos operadores modais de necessidade da linguagem pode ser distribuído sobre a implicação.

1. $K_i(\varphi \rightarrow \psi) \rightarrow K_i\varphi \rightarrow K_i\psi$ para i=1, 2 ... m.
2. $[v_i](\varphi \rightarrow \psi) \rightarrow [v_i]\varphi \rightarrow [v_i]\psi$ para i=1, 2 ... m.
3. $\Box_i(\varphi \rightarrow \psi) \rightarrow \Box_i\varphi \rightarrow \Box_i\psi$ para i=1, 2 ... m.
4. $\Box(\varphi \rightarrow \psi) \rightarrow \Box\varphi \rightarrow \Box\psi$ para i=1, 2 ... m.

5.4.5.2 Duais

As fórmulas 5 a 8 apresentam as modalidades duais:

5. $B_i\varphi \leftrightarrow \neg K_i\neg\varphi$
6. $<v_i>\varphi \leftrightarrow \neg[v_i]\neg\varphi$
7. $\Diamond\varphi \leftrightarrow \neg\Box\neg\varphi$
8. $\Diamond_i\varphi \leftrightarrow \neg\Box_i\neg\varphi$

5.4.5.3 Hierarquia temporal

As fórmulas 9 a 13 apresentam a hierarquia entre os operadores modais que lidam com tempo na linguagem.

9. $\Box\varphi \rightarrow \Box_i\varphi$ para algum i=1, 2 ... m.
10. $\Box_i\varphi \rightarrow (v_i)\varphi$ para algum evento v_i para i=1, 2 ... m.
11. $\Box\varphi \rightarrow (v_i)\varphi$ para algum evento v_i para i=1, 2 ... m.
12. $(v_i)\varphi \rightarrow <v_i>\varphi$ para algum evento v_i para i=1, 2 ... m.
13. $<v_i>\varphi \rightarrow (v_j)\varphi$ para algum evento v_j para i, j=1, 2 ... m.

A fórmula 9 indica que se φ sempre vale no próximo estado global não importando qual ele seja, então φ vale após a ocorrência do primeiro evento para algum dos agentes do grupo. Na verdade, φ necessariamente vale para todos os eventos que poderiam modificar o estado global atual, mas isso é mais forte do que o que a fórmula 9 expressa. A fórmula 10 diz que se φ vale após a ocorrência do primeiro evento para o agente i, então existe algum evento v_i (que será o primeiro evento) para o qual φ vale. A fórmula 11 é conseqüência de 9 e 10: ela diz que se φ vale no próximo estado global futuro, então φ vale após algum evento v_i, que corresponderia ao avento no futuro após o qual φ se tornou válida.

A fórmula 12 é conseqüência da definição dos operadores envolvidos: se φ vale após o evento v_i, então ela eventualmente vale após v_i. E a13 lida com eventos

ocorridos para dois agentes independentes (que podem até ser o mesmo). Ela atesta que se φ eventualmente vale após v_i, então φ tornou-se ou tornar-se-á válida após algum determinado evento v_j.

5.4.5.4 Conhecimento e tempo

As fórmulas 14 a 17 mostram como o conhecimento pode se propaga no tempo para cada um dos operadores temporais da linguagem.

14. $K_i\Box_i\varphi \rightarrow \Box_i K_i\varphi$ para i=1, 2 ... m.
15. $K_i[v_i]\varphi \rightarrow [v_i]K_i\varphi$ para i=1, 2 ... m.
16. $K_i(v_i)\varphi \rightarrow (v_i)K_i\varphi$ para i=1, 2 ... m.

A fórmula 14 faz com que o conhecimento individual de um agente acerca do tempo seja conservado: se um agente sabe que φ vale no próximo estado global, então no próximo estado global ele saberá φ. As fórmulas 15 e 16 são análogas à 14, sendo que lidam com um evento futuro e não com um estado global futuro.

5.4.5.5 Regras de inferência

1. Modus Ponens: De φ e $\varphi \rightarrow \psi$ derive ψ
2. Generalização de $\Box$ (sempre no futuro): De $\models \varphi$ derive $\Box\varphi$
3. Generalização de $\Box_i$: De $\models \varphi$ derive $\Box_i\varphi$
4. Generalização de $[v_i]$: De $\models \varphi$ derive $[v_i]\varphi$

5.6 Poder de expressão da lógica

Para ilustrar o que podemos expressar através da lógica de conhecimento e eventos para sistemas assíncronos que foi proposta examinaremos a seguir dois sistemas assíncronos com múltiplos agentes, ambos já descritos em detalhes em capítulos anteriores. Para cada exemplo mostraremos a linguagem sendo utilizada para representar o conhecimento envolvido no sistema no decorrer de uma execução.

5.6.1 Exemplo 1: Algoritmo para propagação da informação com realimentação (PIF)

Como primeiro exemplo, examinaremos um sistema com três agentes rodando o algoritmo distribuído para propagação da informação com realimentação (PIF) apresentado no capítulo 3.

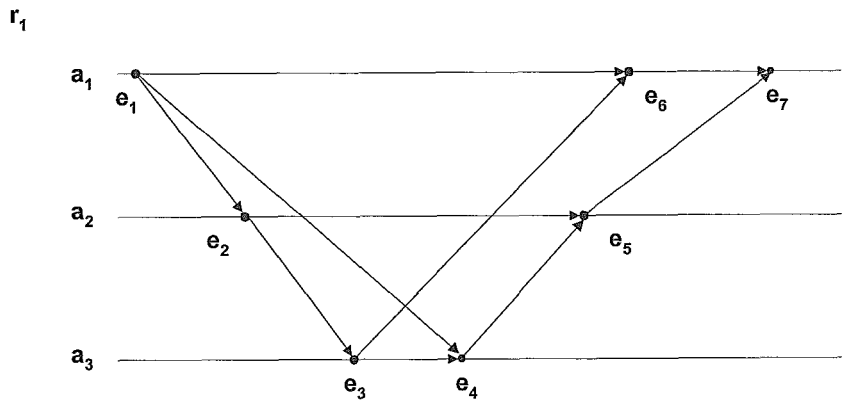

Figura 5.12: Grafo de precedência para uma das possíveis execuções de PIF em um sistema distribuído assíncrono com três agentes.

Seguindo a execução r_1 representada na figura 5.12, temos que depois que o agente a_1 envia a mensagem φ para a_2 e para a_3 (o que ele faz no evento e_1) ele sabe que a_2 e a_3 no futuro saberão φ.

$$K_1 (<e_1>(K_2\varphi \wedge K_3\varphi))$$

Considerando que os agentes a_2 e a_3 só iniciam sua computação após receber a mensagem inicial de a_1 temos que após a ocorrência de e_1 só haverá mudança de estado quando a_2 ou a_3 receber a mensagem de a_1. Logo, depois da ocorrência do próximo evento um entre a_2 ou a_3 saberá φ.

Após a ocorrência de e_1 vale:

$$\Box (K_2\varphi \vee K_3\varphi)$$

$$\Diamond K_2\varphi \qquad\qquad \Diamond K_3\varphi$$

Ao receber a mensagem de a_1, a_2 passa a saber que a1 sabe φ:

$$[e_2] K_2K_1\varphi$$

O mesmo ocorre para o agentes a_3:

$$[e_3] K_3K_2\varphi$$

E ainda para os agentes a_2 e a_3 após e_5 e e_4 respectivamente:

$$[e_5] K_2K_3\varphi \qquad [e_4] K_3K_1\varphi$$

Após e_2, o agente a_2 repassa a mensagem para a_3, logo fica ciente que a_3 saberá φ no futuro.

$$K_2 (<e_2> K_3\varphi)$$

O mesmo ocorre quando a_3 repassa a mensagem a a_1 após e_3.

$$K_3 (<e_3> K_1\varphi)$$

De forma análoga ao que foi feito anteriormente, teremos:

$$[e_6] K_1K_3\varphi \qquad [e_7] K_1K_2\varphi$$

E também:

$$[e_6] K_1K_3 K_1\varphi \qquad [e_7] K_1K_2 K_1\varphi$$

5.6.2 Exemplo 2: Algoritmo distribuído assíncrono para o problema das crianças com lama na testa

Vejamos outro exemplo de aplicação da lógica. Vamos utilizá-la para modelar uma versão assíncrona do problema das crianças com lama na testa. Nesta versão não vale a suposição que todas as crianças escutam as perguntas do pai simultaneamente, pois toda a comunicação em um sistema distribuído assíncrono é feita através de trocas de mensagem, sendo que o tempo de entrega das mensagens é finito porém indeterminado e pode variar para cada mensagem no sistema, o que implica que não há sinal de sincronismo no sistema.

Para que o sistema tenha algum referencial de tempo global é preciso que cada agente simule o sincronismo através de feedback dos outros agentes. Dessa forma, após o recebimento de uma mensagem do pai, os agentes correspondentes a cada uma das crianças enviam uma mensagem de resposta de volta ao pai, contendo a sua resposta à pergunta recebida e indiretamente informando o recebimento da mensagem enviada anteriormente. O pai por sua vez só envia às crianças a próxima mensagem após todas terem recebido a anterior, ou seja, após o recebimento de uma resposta de cada criança, que pode ser afirmativa ou negativa.

Lembrando que agora as repostas de uma criança não podem ser ouvidas por todos os outros membros do grupo (visto que ela só é enviada ao pai) incluímos no modelo assíncrono as seguintes premissas: o pai pára de fazer perguntas quando recebe a primeira resposta afirmativa do sistema. Após este evento, ele envia um sinal de finalização a todos os membros do grupo. Isto significa que se um agente recebe uma pergunta do pai, então todas as crianças responderam negativamente a todas as perguntas anteriores.

Esse processo bastante simples controla as defasagens de tempo entre os agentes. Considerando que o intervalo que compreende cada pergunta do pai e o recebimento de todas as respectivas respostas caracteriza um turno na computação, as

restrições acima garantem que nenhum dois agentes do grupo estejam simultaneamente defasados de mais de um turno .

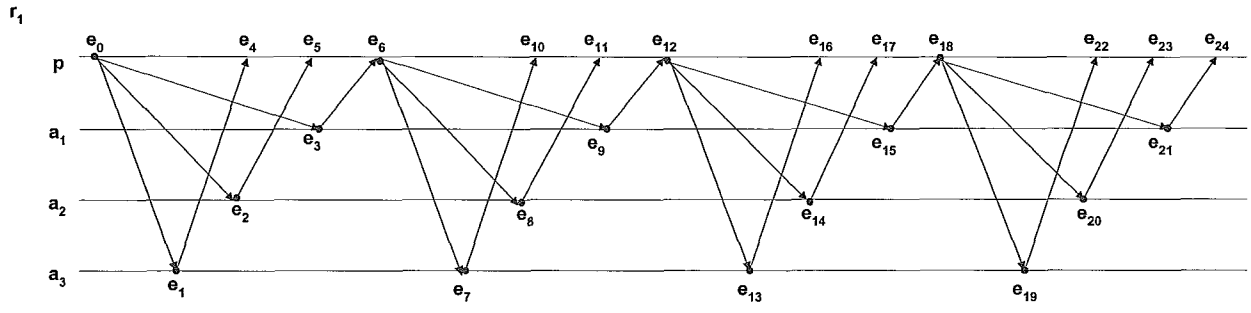

Figura 5.13: Grafo de precedência para uma execução do algoritmo distribuído assíncrono do problema das crianças com lama na testa. Esta é uma das possíveis execuções correspondentes a uma configuração inicial onde as três crianças estão sujas.

Vamos considerar um sistema com três crianças onde todas estão sujas, conforme representado na figura 5.13. O agente p corresponde ao pai e a_1, a_2 e a_3 representam as três crianças do grupo. O evento e_0 corresponde ao anúncio inicial feito pelo pai de que há pelo menos uma criança com lama na testa. Os eventos e_1, e_2 e e_3, são as mensagens que indicam que as crianças receberam o anúncio inicial, e e_4, e_5 e e_6 indicam o recebimento dessas mensagens pelo pai. A partir de então, começam os turnos de perguntas feitas pelo pai. Essas perguntas ocorrem nos eventos e_6, e_{12} e e_{18}. Os demais eventos indicam recebimento das perguntas do pai e as respectivas mensagens de resposta. As respostas das crianças são sempre negativas para todas as perguntas menos para a última, que é disparada no evento e_{18}. Para esta pergunta, as crianças sujas responderão positivamente.

Vejamos agora o que podemos expressar na linguagem. Considerando o seguinte estado de conhecimento inicial de para os agentes, incluindo também que as informações gerais acerca do modo de funcionamento do sistema são de conhecimento comum no grupo.

Fórmulas válidas no estado inicial:

$$S_1, S_2, S_3 \qquad K_2(S_1 \wedge S_3)$$

$$K_1(S_2 \wedge S_3) \qquad K_3(S_1 \wedge S_2)$$

Depois do primeiro anúncio do pai, as crianças sabem que há pelo menos uma criança suja no grupo:

$(e_1)K_1(S_1 \vee S_2 \vee S_3)$ $\quad$ $(e_1)B_1(\neg S_1 \wedge B_3(\neg S_3 \wedge K_2 S_2))$

$(e_2)K_2(S_1 \vee S_2 \vee S_3)$ $\quad$ $(e_2)B_2(\neg S_2 \wedge B_3(\neg S_3 \wedge K_1 S_1))$

$(e_3)K_3(S_1 \vee S_2 \vee S_3)$ $\quad$ $(e_3)B_3(\neg S_3 \wedge B_2(\neg S_2 \wedge K_1 S_1))$

Além disso, ao receber essa mensagem, cada criança sabe que as outras do grupos eventualmente também receberão a mensagem.

$$<e_1>K_1(K_2(S_1 \vee S_2 \vee S_3) \wedge K_3(S_1 \vee S_2 \vee S_3))$$

$$<e_2>K_2(K_1(S_1 \vee S_2 \vee S_3) \wedge K_3(S_1 \vee S_2 \vee S_3))$$

$$<e_3>K_3(K_1(S_1 \vee S_2 \vee S_3) \wedge K_2(S_1 \vee S_2 \vee S_3))$$

$$<e_1>E_G(S_1 \vee S_2 \vee S_3)$$

Mais especificamente, todos sabem que após e_6 todas as crianças já receberam a primeira mensagem do pai, logo:

$$[e_6]E_G^2(S_1 \vee S_2 \vee S_3)$$

Então se iniciam as repetidas perguntas, e os B's são gradativamente transformados em K's. No primeiro turno, todos respondem negativamente e nada acontece.

$$(e_7)B_1(\neg S_1 \wedge B_3(\neg S_3 \wedge K_2 S_2))$$

$$(e_8)B_2(\neg S_2 \wedge B_3(\neg S_3 \wedge K_1 S_1))$$

$$(e_9)B_3(\neg S_3 \wedge B_2(\neg S_2 \wedge K_1 S_1))$$

No segundo turno iniciado em e_{12}, todos percebem que já se passou um turno e portanto só respostas negativas foram recebidas até então.

$$<e_{12}>E_G((S_1 \wedge S_2) \vee (S_1 \wedge S_3) \vee (S_2 \wedge S_3))$$

Agora, alguns B's são transformados em K's:

$$(e_{13})B_1(\neg S_1 \wedge K_3(\neg S_3 \wedge K_2 S_2))$$

$$(e_{14})B_2(\neg S_2 \wedge K_3(\neg S_3 \wedge K_1 S_1))$$

$$(e_{15})B_3(\neg S_3 \wedge K_2(\neg S_2 \wedge K_1 S_1))$$

E cada agente individualmente:

$$<e_{13}> K_1 E_G((S_1 \wedge S_2) \vee (S_1 \wedge S_3) \vee (S_2 \wedge S_3))$$

$$<e_{14}> K_2 E_G((S_1 \wedge S_2) \vee (S_1 \wedge S_3) \vee (S_2 \wedge S_3))$$

$$<e_{15}> K_3 E_G((S_1 \wedge S_2) \vee (S_1 \wedge S_3) \vee (S_2 \wedge S_3))$$

No terceiro turno, todos sabem que só respostas foram recebidas no turno anterior, e novos B's são transformados em K's.

$$<e_{18}>E_G(S_1 \wedge S_2 \wedge S_3)$$

$$(e_{19})K_1\ (S_1 \wedge S_2 \wedge S_3)$$

$$(e_{20})K_2\ (S_1 \wedge S_2 \wedge S_3)$$

$$(e_{21})K_3\ (S_1 \wedge S_2 \wedge S_3)$$

Para cada agente:

$$<e_{19}>E_G\ (S_1 \wedge S_2 \wedge S_3)$$

$$<e_{20}>\ E_G\ (S_1 \wedge S_2 \wedge S_3)$$

$$<e_{21}>\ E_G\ (S_1 \wedge S_2 \wedge S_3)$$

Porém, uma criança não sabe o momento em que as outras crianças receberão a mensagem com a terceira pergunta, logo não conseguimos estabelecer $E_G^2(S_1 \wedge S_2 \wedge S_3)$, pois para isso cada criança teria de receber uma notificação de que as outras já receberam as mensagens anteriores. Para obter E_G^3 um segundo turno de confirmações deveria acontecer, e assim por diante. Só seria possível atingir E_G^k após um número k de turnos, e C_G não pode ser obtido porque não há simultaneidade em sistemas assíncronos. Em contrapartida, o Conhecimento Comum Concorrente de $S_1 \wedge S_2 \wedge S_3$ pode ser obtido ao final do sistema.

Como pudemos observar através dos exemplos, a Lógica de Conhecimento e Eventos é um formalismo e capaz de modelar conhecimento e tempo em sistemas distribuídos assíncronos, adequado ao modelo baseado em eventos apresentado em BARBOSA (1996) para sistemas distribuídos assíncronos. Há vários operadores temporais, o que dá à linguagem alto poder de expressão temporal, mesmo considerando o tempo através de ordenações parciais sobre os eventos. O modelo de Kripke para suportar conhecimento torna a Lógica de Conhecimento e Eventos compatível com toda gama de operadores de conhecimento compatível com o conceito de mundos possíveis, desde que estes sejam também passíveis de utilização em um sistema assíncrono, o que não é o caso do operador para conhecimento comum definido em HALPERN & MOSES (1984).

Capítulo 6

Conclusões

O assunto desta Tese foi o tratamento do conhecimento e do tempo em sistemas distribuídos através de lógicas, enfatizando os sistemas distribuídos assíncronos. Várias lógicas foram revistas e duas foram apresentadas, uma para sistemas síncronos com modalidades de ação e outra baseada no modelo fortemente assíncrono sem falhas para sistemas distribuídos proposto em BARBOSA (1996).

Além da restrição de ausência de falhas, tal modelo não impõe nenhuma outra restrição em relação ao comportamento temporal do sistema ou ao comportamento temporal de cada agente, o que torna a nova linguagem livre de restrições temporais, capaz de modelar qualquer tipo de algoritmo distribuído assíncrono compatível com o modelo genérico de tarefa reativa de BARBOSA.

Em especial dois problemas distintos envolvendo conhecimento em sistemas distribuídos assíncronos foram modelados com a linguagem nova, mostrando seu alto poder de expressão e comprovando sua aplicabilidade.

6.1 Contribuições

Foram estudadas lógicas de conhecimento e tempo para sistemas distribuídos, especialmente as apresentadas nos trabalhos de FAGIN *et al.* (1995) e LEHMANN (1984). Constatou-se que ambas aplicam-se muito bem à classe de sistemas distribuídos síncronos, porém nenhuma seria compatível com a grande flexibilidade temporal do modelo para sistemas distribuídos assíncronos de BARBOSA (1996).

Após detalhado estudo do modelo de BARBOSA, criamos uma lógica de conhecimento e tempo onde o tempo foi tratado através de modalidades de ação. Tal trabalho, objeto de um artigo apresentado no XXI Congresso da Sociedade Brasileira de Computação - Encontro Nacional de Inteligência Artificial - DELGADO e BENEVIDES (2001) - baseava-se ainda nas lógicas FAGIN *et al.* e LEHMANN supracitadas, e aplicava-se a sistemas distribuídos síncronos.

Como evolução deste primeiro trabalho, e principal contribuição desta tese, temos a apresentação de uma lógica para o tratamento de conhecimento e tempo em sistemas distribuídos assíncronos. Nesta lógica o tratamento do tempo é feito através de um formalismo baseado em eventos, de forma que a linguagem esteja apta a modelar o conhecimento em sistemas distribuídos descritos segundo o modelo fortemente assíncrono apresentado em BARBOSA (1996). Novos operadores temporais baseados em eventos foram definidos, dando à linguagem alto poder de expressão temporal mesmo considerando o tempo através de ordenações parciais sobre os eventos. O modelo de Kripke para suportar conhecimento torna a Lógica de Conhecimento e Eventos compatível com toda gama de operadores de conhecimento baseados no conceito intuitivo de mundos possíveis, desde que estes sejam também passíveis de utilização em um sistema assíncrono, o que não é o caso do operador para conhecimento comum definido em HALPERN & MOSES (1984). Outras noções de conhecimento de grupo compatíveis com o modelo assíncrono foram citadas, em especial a noção de Conhecimento Comum Concorrente definida em PANANGADEN & TAYLOR (1992), em substituição ao Conhecimento Comum para os sistemas assíncronos.

6.2 Direções de pesquisa para trabalhos futuros

A lógica apresentada representa um novo tratamento para conhecimento e tempo em sistemas distribuídos assíncronos, com bom poder de aplicabilidade. O modelo apresentado nesta tese deixa em aberto uma série de aspectos de interesse. Apresentamos aqui seqüências naturais do trabalho, e também sugestões para aplicação do modelo em outras áreas.

6.2.1 Apresentação de um sistema axiomático e provas de corretude e completude

Para concluir a formalização do modelo, o próximo passo seria a apresentação de um sistema axiomático. Este passo já foi iniciado neste trabalho, na seção 5.4.5 do capítulo 5. Cabe agora investigar o conjunto de fórmulas válidas, definir o sistema axiomático e elaborar as provas de corretude e completude para a nova linguagem.

6.2.2 Model Check

A lógica de conhecimento e eventos é bastante adequada para expressar propriedades de sistemas distribuídos, de forma que dado um algoritmo distribuído, utilizando *Model Check* poderíamos ser capazes de verificar a validade de fórmulas no modelo e com isso provar as propriedades do algoritmo implementado. Um futuro trabalho de implementação seria construir uma forma canônica para especificação de algoritmos distribuídos e da lógica de conhecimento e eventos, de forma que dado um algoritmo distribuído descrito segundo o modelo apresentado em BARBOSA (1996) fosse possível obter o *frame* correspondente para a lógica, incluindo os conjuntos de eventos, relações entre os cortes consistentes e valorações nos mundos possíveis. Tal *frame* seria utilizado para a aplicação de *model check*.

6.2.3 Provador de Teorema

Uma vez apresentada a axiomatização e o conjunto de regras de inferência, gostaria de estudar a questão de decidibilidade e conseqüentemente propor um provador automático para a Lógica de conhecimento e Eventos.

6.2.4 Exemplos de Aplicação

A aplicação da lógica foi explorada apenas no contexto de exemplos básicos de propagação do conhecimento em sistemas distribuídos assíncronos. Seria de grande expressão a aplicação da lógica de conhecimento e eventos a outras classes de problemas de sistemas distribuídos assíncronos, como especificação de protocolos de trocas de informação elaborados, como por exemplo o TCP/IP. Um algoritmo baseado em conhecimento para este protocolo foi proposto em STULP & VERBRUGGE (2000), de forma que outros algoritmos baseados em conhecimento poderiam ser escritos para este tipo de aplicação, tornando-as passíveis de ser descritas através da lógica de conhecimento e eventos.

6.2.5 Inclusão de Operadores para representar conhecimento comum na linguagem

A lógica de conhecimento e eventos não possui até o momento nenhum operador para representar conhecimento de grupo, em especial o conhecimento comum. Há interesse em definir utilizando os outros operadores da lógica um conjunto de operadores capaz de representar o conhecimento do grupo de agentes envolvido no

sistema, em especial um operador para conhecimento comum no estilo do operador para conhecimento comum concorrente de PANAGADEN & TAYLOR (1992) deve ser apresentado.

Referências Bibliográficas

BARBOSA, V. C., 1996, *An Introduction to distributed Algorithms*, 1ª edição, Massachusetts, EUA, MIT Press.

BLACKBURN, P., RIJKE, M., VENEMA, Y., 1998, *Modal Logic*, ILLC Amsterdam & computer linguistics, Saarbruecken.

COSTA, V., 2001, U*ma Lógica Bidimensional para Conhecimento em Sistemas Distribuídos.* Tese de D.Sc., COPPE/UFRJ, Rio de Janeiro, RJ, Brasil.

DELGADO, C., BENEVIDES, M., 2001, "Dynamic Knowledge Logic". *Anais do XXI Congresso da Sociedade Brasileira de Computação*, v. 1, pp. 187-197, Fortaleza - CE, Julho.

DOLEV, D., MOSES, Y., HALPERN, J. Y., 1985, *Cheating Husbands and Other Stories: a Case Study of Knowledge, Action and Communication*, Relatório de Pesquisa da IBM, RJ 4756 (50524).

FAGIN, R., HALPERN, J. Y., MOSES, Y. et al., 1995, *Reasoning About Knowledge*, 1ª edição, Massachusetts, EUA, MIT Press.

HALPERN, J. Y., MOSES, Y., 1990, "Knowledge and Common Knowledge in a Distributed Environment", *Journal of the ACM*, v. 37, n. 3.

HUGHES, G. E., CRESSWELL, M. J., 1996, *A New Introduction to Modal Logic*, Londres, Inglaterra.

KRIPKE, S., 1959, "A Completeness Theorem in Modal Logic", *Journal of Symbolic Logic*, 24 (1), Março.

KRIPKE, S., 1963, "Semantical Considerations in Modal Logics", *Acta Philosophica Fennica*, 16.

LAMPORT, L., 1978, "Time, clocks, and the ordering of events in a distributed system", *Comm. of the ACM*, 21, pp. 558-565.

LAMPORT, L., 1985, "Paradigms for Distributed Computing". Paul M, Siegert HJ (eds) *Methods and tools for especification, an advanced course*, Lect Notes Comput Sci, vol 190. Springer, Berlin Heidelberg New Yourk, pp. 19-30, 454-468.

LEHMANN, D., SHELAH, S., 1982, "Reasoning with Time and Chance", *Information and Control* v. 53, pp. 165-198.

LEHMANN, D. J., 1984, "Knowledge, Common Knowledge and related puzzles (Extended Summary)". *Proceedings of the Third Annual ACM Symposium on Principles of Distributed Computing*, pp 62-67, Vancouver, B. C., Canada, Agosto.

PANANGADEN, P., TAYLOR, K., 1992, "Concurrent Common Knowledge: Defining Agreement for Asynchronous Systems", *Distributed Computing*, 6(2), pp 73-93.

STULP, F., VERBRUGGE, R., 2000, "A knowledge-based algorithm for the Internet protocol TCP", *Proceedings of the 4rd Conference on Logic and the Foundations of Game and Descision Theory (LOFT 4)*, Torino, Italy, June.