MODELAGEM E VERIFICAÇÃO DE PROPRIEDADES EPISTÊMICAS EM SISTEMAS MULTI-AGENTES

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 DOUTOR EM CIÊNCIAS EM ENGENHARIA DE SISTEMAS E COMPUTAÇÃO.

Aprovada por:

Prof. Mario Roberto Folhadela Benevides, Ph.D.

Prof. Valmir Carneiro Barbosa, Ph.D.

Prof. Gerson Zaverucha, Ph.D.

Prof. Edward Hermann Haeusler, D.Sc.

Prof. Marcelo Finger, Ph.D.

RIO DE JANEIRO, RJ - BRASIL

NOVEMBRO DE 2007

DELGADO, CARLA AMOR DIVINO MOREIRA

Modelagem e Verificação de Propriedades Epistêmicas em Sistemas Multi-Agentes [Rio de Janeiro] 2007

IX, 135 p. 29,7-cm (COPPE/UFRJ, D.Sc., Engenharia de Sistemas e Computação, 2007)

Tese – Universidade Federal do Rio de Janeiro, COPPE

1. Sistemas multi-agentes
2. Modelos probabilísticos
3. Lógica modal de conhecimento e crença
4. Lógica temporal probabilística
5. Verificação automática de modelos

I. COPPE/UFRJ II. Título (série)

AGRADECIMENTOS

Agradeço ao Prof. Mario Benevides, por sua orientação, seu apoio e incentivo inconteste ao meu trabalho.

Agradeço aos professores Colin Stirling e Prof. Julian Bradfield, e ao Laboratory for Foundations of Computer Science, pelo suporte a minha estadia na Universidade de Edimburgo.

Agradeço a todos os colegas do programa, entre professores, alunos e funcionários, que direta ou indiretamente contribuíram para que essa Tese fosse realizada. Em especial, aos professores Sheila e Paulo Veloso, e aos (ex-)alunos Michel Carlini, Kate Revoredo, Flavia Cruz, Renata Freitas e Petrucio Viana.

Agradeço a Geraldo Zimbrão pelo apoio.

Agradeço a José Xexeo pelo conselho certo na hora certa.

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

Agradeço a Deus, por continuar me abençoando com oportunidades.

Dedico este trabalho ao meu pai, Adilson Delgado.

Parcialmente financiado pela CAPES e pelo CNPQ.

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

MODELAGEM E VERIFICAÇÃO DE PROPRIEDADES EPISTÊMICAS EM SISTEMAS MULTI-AGENTES

Carla Amor Divino Moreira Delgado

Novembro/2007

Orientador : Mario Roberto Folhadela Benevides
Programa : Engenharia de Sistemas e Computação

O objetivo deste trabalho é desenvolver modelos, estratégias e algoritmos para verificar formalmente propriedades epistêmicas em sistemas Multi-Agentes. Três tipos de modelos são abordados e sua adequação e aplicabilidade discutidas. Primeiro, discutimos verificação de propriedades epistêmicas em modelos não probabilísticos para sistemas multi-agentes. O trabalho se beneficia dos vários resultados recentes na área de verificação de modelos e de lógicas modais combinando operadores de conhecimento e tempo, e resulta numa proposta para construir as relações de conhecimento diretamente a partir de especificações de sistemas baseadas em autômatos. Estendendo a discussão para sistemas probabilísticos baseados em modelos de processos de Markov, consideramos modelos envolvendo probabilidade e não determinismo. As respectivas propostas para construção das relações de conhecimento levam em conta a natureza temporal, probabilística e não determinística dos sistemas, considerando a influência destes aspectos na semântica para conhecimento. Apresentamos para cada modelo uma linguagem para construir especificações envolvendo conhecimento e os correspondentes algoritmos para verificação automática de propriedades.

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

MODEL AND VERIFICATION OF EPISTEMIC PROPERTIES IN MULTI-AGENT SYSTEMS

Carla Amor Divino Moreira Delgado

November/2007

Advisor : Mario Roberto Folhadela Benevides
Department : Computing and Systems Engineering

Our aim is to develop models, languages and algorithms to formally verify knowledge properties of concurrent Multi-Agents Systems. Three models are proposed and their adequacy is discussed with respect to representation and verification of knowledge properties. First, we address the issue of model checking knowledge in non-probabilistic concurrent systems. The work benefits from many recent results on model checking and combined logics for time and knowledge, and focus on the way knowledge relations can be captured from automata-based system specifications. In order to extend the results to probabilistic systems, we use Markov process models involving probability and non-determinism. The corresponding approaches to construct the knowledge relations take into account the temporal, probabilistic and non-deterministic nature of the systems, and we discuss for each case how these aspects influence the semantics for knowledge. For each model we present a language able to express specifications on the knowledge level and the corresponding model checking algorithms.

Sumário

Capítulo 1

Introdução

A crescente importância de sistemas interativos compostos por vários agentes autônomos (sistemas multi-agentes) tornam necessárias novas técnicas de modelagem e verificação de software adequadas a esta classe de sistemas.

O processo de verificação automática de modelos (Model Check) é uma técnica madura e poderosa de verificação de sistemas concorrentes [CE81], e muitos esforços tem sido feitos no sentido de estender o ferramental de verificação de modelos com artefatos específico capazes de contemplar as características dos processos de interação racional que caracterizam os sistemas multi-agentes.

Desde 1980 lógicas de conhecimento vem sendo estudadas e empregadas em especificações formais. Lógicas temporais com operadores modais de conhecimento adicionados também vem sendo amplamente utilizadas para raciocinar sobre sistemas multi-agentes [HV89], [vdHW02a]. As técnicas de verificação de modelos para lógicas temporais e epistêmicas foram estudadas em proporção bem menor que cada um destes dois assuntos (checagem de modelos; representação de conhecimento) em separado, porém algumas iniciativas despertam a atenção para a relevância deste tema [vdMS99], [KLP04], [vdHW02a], [LR06].

As vantagens que esta abordagem proporciona são a oportunidade de modelar evolução do conhecimento no sistema, o que é bastante útil para compreender (e antever) o comportamento do sistema, bem como seu propósito. Através da modelagem do conhecimento envolvido no sistema por parte de cada agente, abrimos espaço para que outras questões como autonomia e interação, duas características marcantes de sistemas concorrentes, possam também ser discutidas.

Em muitas das áreas onde raciocínio sobre conhecimento mostra-se útil, é importante também raciocinar sobre a probabilidade de certos eventos, além do conhecimento dos agentes. Um exemplo de aplicação são sistemas distribuídos onde os programas podem ter componentes aleatórias ou probabilísticas. Probabilidade representa atualmente um papel importante na modelagem e análise de sistemas de software e hardware, incluindo sistemas distribuídos. Apesar de que os métodos de verificação automática de modelos não probabilísticos já estão maduros e sua aplicação difundida, os processos de verificação probabilística de modelos ainda podem ser considerados como estando em seus estágios iniciais de desenvolvimento.

O foco desta tese é o processo de modelagem composicional, especificação e verificação de sistemas multi-agentes no nível do conhecimento. Apresentamos nos capítulos seguintes uma revisão das classes de modelos adequados para métodos de verificação automática, e discutimos as atuais abordagens para obter especificações composicionais para sistemas multi-agentes. Mostraremos também as principais linguagens já propostas para raciocinar sobre conhecimento, tempo, probabilidade, e algumas que equacionam duas destas classes de propriedades. Considerando o referencial bibliográfico e trabalhos correlatos, podemos dizer que as técnicas de verificação de modelos estão bem estabelecidas para sistemas discretos e em evolução para sistemas probabilísticos. Ao incorporar conhecimento entre as propriedades que se deseja raciocinar, existem contribuições relevantes de verificação de propriedades epistêmicas e temporais em modelos não probabilísticos, porém poucas efetivamente já aplicadas. Até o presente momento a autora desconhece contribuições na área de verificação de propriedades epistêmicas e temporais em modelos probabilísticos.

1.1 Contexto bibliográfico

O trabalho de pesquisa desta tese envolve os seguintes temas principais: conhecimento em sistemas multi-agentes, verificação de sistemas, verificação de sistemas probabilísticos. Embora estes temas sejam abordados nos capítulos 2 e 3, discutimos a seguir brevemente a relevância destes assuntos e as inter-relações entre eles.

1.1.1 Conhecimento em sistemas multi-agentes

Vários sistemas modernos são contruídos sobre uma arquitetura multi-agentes. Um sistema multi-agentes é composto de partes autônomas e configuráveis, que chamamos de agentes. A vantagem da arquitetura multi-agentes é a flexibilidade, uma vez que os agentes podem ser implementados e modificados sem grande impacto no sistema, bem como reutilizados em outros sistemas. Os sistemas multi-agentes tendem a ser robustos o suficiente para recuperarem-se de falhas, dadas suas características de auto-gerenciamento [BS00]. Tais características são conseqüência de dois fatores principais: autonomia e interação racional. Quanto mais autônomos e capazes de interagir de forma eficiente sejam os agentes, mais robusto é o sistema multi-agente.

Historicamente, conhecimento é um conceito altamente associado a autonomia e interação. Falar sobre sistemas multi-agentes em termos de conhecimento tem se mostrado útil em diversas áreas como por exemplo computação distribuída, inteligência artificial ou economia [Hal91]. Em teoria dos jogos, por exemplo, a estratégia de um jogador tipicamente leva em conta o conhecimento que o jogador adquire sobre as estratégias dos outros jogadores, o que se dá usualmente através de interação. Intuitivamente, quanto mais conhecimento tem o agente, mais eficiente será o seu comportamento, pois ele saberá usar de sua autonomia para escolher as ações que serão mais efetivas dado seu conhecimento do sistema.

As relações entre conhecimento e probabilidade foram abordadas por alguns autores [FH94, FZ88, HMT88, Hal91]. Os resultados mais interessantes são modelos abstratos para conhecimento e probabilidade onde se atribui a cada par agente–estado um espaço de probabilidade. Tais espaços de probabilidade são usados para computar a probabilidade, segundo o ponto de vista do agente, de que a fórmula ϕ seja verdadeira em um estado do sistema. Uma parte importante do problema de modelar conhecimento e probabilidade é escolher a atribuição de espaços de probabilidade, uma vez que a escolha depende das características do modelo.

1.1.2 Verificação de sistemas

Existe um grande interesse em verificar se um sistema multi-agentes executa da forma desejada, ou se atinge certos objetivos. O processo de verificação automática de modelos

(*model check*) é uma técnica bem sucedida para estabelecer a corretude do modelo, em relação a um dado conjunto de propriedades da lógica temporal que o sistema deve satisfazer [PCG00]. Usando técnicas eficientes, o processo de verificação de modelos tem sido aplicado a modelos de escala industrial envolvendo mais de 10^{30} estados [Her01]. Levando em conta que uma das característica de sistemas multi-agentes é que estes consistem numa hierarquia de componentes interativas porém autônomas, e dada a crescente complexidade destes sistemas, uma análise composicional condizente com a arquitetura multi-agentes seria adequada [Her01]. Alguns formalismos vem sendo desenvolvidos para modelar tais sistemas baseados no princípio da composicionalidade.

Outro aspecto a ser explorado no processo de modelagem e verificação de sistemas multi-agentes são os aspectos epistêmicos do sistema. Existem formalismos para modelar conhecimento e especificar propriedades epistêmicas, dos quais o mais conhecido são as lógicas modais de conhecimento e crença. Inclusive, o processo de modelagem e verificação tradicional foi estendido para propriedades epistêmicas em alguns trabalhos recentes [vdMS99, KLP04, vdHW02a, LR06]. Esta tese também apresenta uma contribuição nesta área, estendendo lógicas temporais probabilísticas com modalidades de conhecimento.

Podemos considerar que um processo de verificação automática adequado a sistemas multi-agentes deveria permitir a construção do modelo de forma composicional, a partir de modelos contruídos para cada agente, e viabilizar a verificação de propriedades epistêmicas, além das propriedades convencionais relacionadas a evolução. Abordagens deste tipo foram feitas para sistemas não probabilísticos, porém contribuições nesta área para sistemas probabilísticos ainda são poucas.

1.1.3 Verificação de sistemas probabilísticos

Assim como em várias áreas de computação, técnicas probabilísticas vem sendo cada vez mais utilizadas em sistemas multi-agentes – para aumentar a eficiência (ao estilo de algoritmos aleatórios), para quebra de simetria ou como um artefato inerente ao problema modelado (tolerância a falhas) [MM04]. Com o crescente interesse nesta área, estimulado por suas diversas aplicações, estudos para aumentar a compreensão das relações entre comportamento autônomo, probabilidade e interação vêm sendo feitos ([RKNP04, KKNP01, Her02], para mencionar alguns exemplos). Ao considerar esta classe de sis-

temas é natural raciocinar, mesmo que informalmente, sobre conhecimento e probabilidade na evolução do sistema. Por exemplo, o sistema pode ser feito de forma que se um agente sabe que a probabilidade de que uma ação de comunicação falhe é menor que $99,98$ por cento, ele pode considerar relaxar um pouco os mecanismos de redundância no protocolo de comunicação.

Em comum com o processo convencional de verificação de modelos, a verificação probabilística de modelos envolve a análise de estados alcançáveis em um sistema de transição de estados, e além disso incorpora o cálculo de estimativas através de métodos numéricos ou analíticos. Enquanto processos de checagem de modelos convencionais trabalham com modelos de estados e transições e especificações feitas em lógicas temporais, na verificação probabilística os modelos considerados são variantes de cadeias de Markov, onde além de estados e transições estão também codificadas as probabilidades de que uma transição seja feita [RKNP04].

Cadeias de Markov são amplamente usadas como modelos estocásticos de fenômenos do mundo real, principalmente porque possuem uma propriedade que simplifica tanto a modelagem quanto a análise do modelo – a propriedade de "ausência de memória". Tal propriedade representa que a evolução futura de uma cadeia de Markov é independente do passado, e depende apenas no estado corrente. Cadeias de Markov de tempo discreto são convenientes para descrever a evolução estocástica de sistemas seqüenciais. As transições que emanam de cada estado definem como a massa de probabilidade será distribuída no próximo instante de tempo. Como estas cadeias evoluem em espaços discretos de tempo, o fluxo de probabilidades não é contínuo, e o sistema evolui em saltos durante os quais o fluxo de probabilidade permanece inalterado (por exemplo entre o instante 2 e o instante 3). Isto é conveniente para sistemas seqüenciais. Cadeias de Markov de tempo contínuo são interpretadas sobre tempo contínuo, amplamente utilizadas para modelar comportamento estocástico de sistemas concorrentes, por aliar simplicidade matemática com conveniência do modelo.

1.2 Trabalhos Correlatos

Algoritmos para verificação de modelos envolvendo conhecimento foram explorados em alguns trabalhos recentes como [BGS99], [EJT98], [vdMS99], [KLP04], [vdHW02a]

e [LR06], para citar alguns.

Estes trabalhos exploram algumas das lógicas definidas para expressar propriedades epistêmicas e temporais desejáveis para modelar protocolos e sistemas distribuídos, entre outras coisas. Os trabalhos mais expressivos nesta área e que mais relações têm com o trabalho realizado nesta tese serão descritos brevemente a seguir.

Outros trabalhos também no escopo da tese abordam lógicas epistêmicas e probabilísticas, dos quais os mais relevantes serão comentados no capítulo 2.

Até o presente momento a autora não encontrou ou teve notícia de trabalhos na área de checagem de modelos probabilísticos envolvendo tempo e conhecimento. Nosso trabalho se beneficia das contribuições apresentadas nos trabalho relacionados investigados, e foca no preenchimento das lacunas para completar o ferramental disponível.

1.2.1 Verificando Conhecimento e Tempo

O trabalho de van der Hoek e Wooldridge [vdHW02a] apresenta um processo de verificação de propriedades expressas em lógica modal temporal e epistêmica sobre modelos para sistemas multi-agentes. A lógica utilizada é CKL_n, proposta por Halpern e Vardi [HV89], e a abordagem apresentada no artigo de Hoek e Wooldridge é uma combinação das idéias de semântica de Sistemas Interpretados [FHM95] para conhecimento com a lógica de proposições locais [EvdMM98] (o modelo de sistemas interpretados foi descrito na seção 3.1.1, definição 3.1.5). O processo de verificação de modelos apresentado é redutível à verificação de Lógicas de Tempo Linear (*Linear Time Logics* – LTL).

Em resumo, o modelo de Sistemas Interpretados ([FHM95]) é pensado para sistemas compostos por múltiplos agentes, cada qual consistindo em um processo independente. Considerando que $Ag = \{1, \cdots, n\}$ representa o conjunto de agentes, L_i representa o conjunto de possíveis estados locais de um agente $i \in Ag$. O estado do sistema em um ponto no tempo é caracterizado por uma tupla $\langle l_1, \cdots, l_n \rangle$, onde $l_i \in L_i$ é o estado local do agente i neste momento. Uma execução (definição 3.1.3) junto com um dado instante no tempo representa um "ponto" – um ponto (r, u) define um estado global $r(u)$. A i-ésima componente de uma tupla $r(u)$ é denotada por $r_i(u)$, que representa o estado local do agente i na execução r e no tempo u. Uma função de valoração π determina o conjunto de proposições primitivas verdadeiras a cada ponto de $\mathcal{R}$.

A abordagem de [FHM95] é utilizada para incorporar conhecimento ao modelo. A cada agente i associa-se uma relação de equivalência $\sim_i$ sobre o conjunto de pontos: $(r, u) \sim_i (r', v)$ se e somente se $r_i(u) = r'_i(v)$. Esta é a relação usada para dar semântica aos operadores de conhecimento de CKL_n.

CKL_né uma lógica proposicional temporal aumentada com modalidades de conhecimento K_i, uma para cada agente i, e operadores de conhecimento comum C_Γ, onde $\Gamma \subseteq Ag$. As fórmulas da linguagem são construídas sobre um conjunto de proposições primitivas, utilizando os conectivos proposicionais $\neg$, $\vee$, $\wedge$, $\rightarrow$, $\leftrightarrow$ e os conectivos temporais $\bigcirc$ (próximo – *next*), $\diamond$ (eventualmente), $\Box$ (sempre no futuro), $\mathcal{U}$ (até – *until*) e $\mathcal{W}$ (até fraco – *weak until*). A semântica para CKL_n é dada através da relação de satisfação "$\models_{CKL_n}$", que vale entre pares da forma $\langle \mathcal{I}, (r, u) \rangle$ e fórmulas de CKL_n, da forma usual.

Se retirarmos os operadores de conhecimento e conhecimento comum de CKL_n, a linguagem obtida é a mesma que LTL. A checagem de modelos para LTL possui a vantagem de que cada fórmula da linguagem são avaliadas levando em conta uma única execução. Os autores gostariam de fazer uso de ferramentas de checagem de modelos já construídas para LTL, como por exemplo SPIN [Hol91], [Hol04] para o processo de verificação de modelos CKL_n, porém há que tratar os operadores de conhecimento, que necessitam levar em conta possivelmente mais de uma execução para serem verificados.

Um processo de redução de CKL_n a LTL é apresentado, onde a idéia principal é definir os operadores de conhecimento em termos de quantificadores da Lógica de Proposições locais (*Logic of Local Propositions* – LLP) de [EvdMM98]. LLP é uma lógica modal com uma única modalidade universal, $\mathtt{Nec}$, e que permite quantificação sobre proposições. A fórmula $\mathtt{Nec}\phi$ significa que ϕ é verdadeiro em todos os estados. LLP possui uma coleção de quantificadores $\forall_i, \exists_i$ (onde i é um agente), que permitem quatificação sobre proposições que são locais a um agente. Intuitivamente, uma proposição é local ao agente i se i é capaz de determinar sua validade usando apenas informação proveniente de seu estado local. Utilizando estes quantificadores, podemos definir: $K_i\phi \doteq \exists_i q[q \wedge \mathtt{Nec}(q \rightarrow \phi)]$. Então o agente i sabe ϕ se e somente se existe uma proposição q local a i tal que q vale, e sempre que q vale, ϕ também vale.

Esta idéia é usada para reduzir uma fórmula envolvendo conhecimento de CKL_n para LTL. Uma função é definida de forma que receba como entrada um sistema interpretado

e uma fórmula ϕ de LTL, e retorne uma proposição local ψ que implique globalmente em ϕ. ψ funciona como uma proposição i-local de ϕ em (r, u), e é construída de forma que: $\langle \mathcal{I}, (r, u) \rangle \models_{CKL_n} K_i\phi$ se e somente se $\langle \mathcal{I}, (r, u) \rangle \models_{LTL} \psi$. Desta forma, é possível utilizar o processo e as ferramentas de checagem de modelos LTL para CKL_n.

O trabalho descrito apresenta uma forma de verificar propriedades epistêmicas em sistemas não probabilísticos de tempo discreto que pode ser implementada fazendo uso do ferramental já disponível para verificação automática de modelos LTL.

1.2.2 Checagem de especificações em lógica temporal e epistêmica para tempo alternado - ATEL

O trabalho de Alur, Henzinger e Kupferman [AHK02] generaliza a lógica temporal de tempo ramificado tradicionalmente usada em checagem de modelos, (CTL), para a Lógica de Tempo Alternado (*Alternating–Time Temporal Logic* – ATL), substituindo os quantificadores de caminho por "modalidades de cooperação" que podem ser utilizadas para falar dos "poderes" que grupos de agente possuem para atingir determinados resultados. Fórmulas ATL são interpretadas sobre estruturas de teoria dos jogos, o que permite, de acordo com os autores, que se modele composições de sistemas abertos. [1].

A fórmula ATL $\langle\langle A \rangle\rangle \psi$ é satisfeita em um estado q se e somente se há uma estratégia vencedora que permita aos agentes no conjunto A escolher suas ações de tal forma que ψ vale no conjunto de estados sucessores de q que resultam da execução da ação escolhida pelos agentes em A, não importando que ações são executadas por agentes que não estão em A.

[dHW03] estende ATL com modalidades de conhecimento. A linguagem estendida, chamada *Alternating-time Temporal Epistemic Logic* – ATEL, tem o poder de expressar propriedades sobre as relações entre conhecimento e poder de um grupo, uma característica interessante para sistemas multi-agentes. Até o presente momento, ATEL ainda não conta com uma semântica aceita em geral, devido às dificuldades em coordenar as modalidades de conhecimento – originalmente estabelecidas sobre sistemas estáveis – e as modalidades de ações de grupo provenientes de ATL – baseadas em modelos de teoria

[1] "Um sistema aberto é um sistema que interage com o ambiente e cujo comportamento depende do estado do sistema bem como do comportamento do ambiente" [AHK02].

dos jogos e especialmente definidas para sistemas abertos.

De um ponto de vista técnico, ATEL acrescenta a ATL modalidades de conhecimento e as respectivas relações epistêmicas de acessibilidade $\sim_a$ para cada agente a. As relações epistêmicas de acessibilidade desempenham um papel importante na semântica das modalidades de conhecimento, mas são ignoradas na hora de levar em conta as estratégias extraídas do modelo que são utilizadas na definição da semântica dos quantificadores temporais e das modalidades temporais. Isto não é razoável porque a existência de relações epistêmicas de acessibilidade significa a presença de incerteza no modelo em relação a mundos possíveis, o que em teoria dos jogos deveria refletir no conjunto de estratégias dos agentes, pois as estratégias que um agente dispõe devem ser uniformes sobre o conjunto de mundos possíveis. As devidas mudanças na definição de estratégia afetariam a semântica, porém não são consideradas pelos autores. Algumas iniciativas para tratar este problema foram apresentadas em [Jon03], [JvdH04], [Ago06]. Em paralelo, abordagens para reduzir o problema de checagem de modelos com ATEL para checagem de modelos com ATL foram discutidas em [vdHW02b], [GJ04], [SO03] e [LR06].

Em particular, [LR06] apresenta uma abordagem interessante para verificação de propriedades epistêmicas, temporais e das propriedades nativas de ATL em modelos de Sistemas Interpretados (definição 3.1.5. Um mapeamento entre os modelos de teoria dos jogos usados em ATL (especificamente, Estruturas de Jogos Concorrentes, *Concurrent Game Structures* – CGS) e Sistemas Interpretados é apresentado e discutido, e uma forma de obter as relações epistêmicas de acessibilidade muito semelhante à forma apresentada em [BDR$^+$06] [2] – considerando que os estados globais em Sistemas Interpretados são compostos de tuplas de estados locais, a relação epistêmica de acessibilidade para um agente i, R_i^K é construída relacionando os estados globais em que o agente i possui o mesmo estado local.

O algoritmo de verificação de modelos apresentado usa a abordagem de checagem explícita de modelos. O módulo específico para o tratamento dos operadores modais de conhecimento K_i , de acordo com a semântica utilizada, percorre os estados relacionados através de R_i^K. A diferença desta para a abordagem apresentada em [BDR$^+$06] reside unicamente no fato de que a primeira constrói o conjunto de estados onde uma fórmula

[2] As contribuições apresentadas em [BDR$^+$06] estão incorporadas nesta tese, na seção 4.1

do tipo $K_i\phi$ é satisfeita a partir dos estados onde $\neg\phi$ vale, e a segunda, a partir dos estados onde ϕ vale. O algoritmo correspondente ao módulo para identificar os estados onde uma fórmula $K_i\phi$ vale (considerando que existe um módulo SAT recursivo que identifica os estados onde ϕ vale), apresentado em [LR06] é mostrado a seguir. O algoritmo mostrado em [BDR$^+$06] é apresentado na seção 4.1.3.

Algorithm 1: SATK(G, ϕ, i)

begin
$X := SAT(\neg\phi)$
$Y := \{g \in G$ tal que $R_i^K(g, g')$ e $g' \in X\}$
return $\neg Y$
end

1.2.3 Verificação de sistemas multi agentes através de checagem ilimitada de modelos

[KLP04] discute checagem ilimitada de modelos de uma linguagem que combina lógica de tempo ramificado (*Computation Tree Logic* – CTL) e lógica de conhecimento, denominada CTLpK (uma extensão da lógica CTLK de [PL03] com operador de passado), e utiliza uma técnica baseada em SAT para melhorar a eficiência dos algoritmos de checagem de modelos. A semântica da linguagem proposta é baseada em sistemas interpretados, assim como nos trabalhos descritos nas seções 1.2.1 e 1.2.2. O método apresentado é uma extensão da técnica de verificação ilimitada de modelos (*Unbounded Model Checking* – UMC) introduzida em [McM02], que consiste em mapear o problema de checagem de modelos (neste caso, uma fórmula de CTLpK) em um problema de satisfatibilidade de uma fórmula proposicional.

UMC explora a caracterização de modalidades básicas em termos de Fórmulas Booleanas Quantificadas (*Quantified Boolean Formulas* – QBF), e os respectivos algoritmos que mapeiam QBF e equações de ponto fixo sobre QBF em fórmulas proposicionais.

Em [KLP04] são apresentados três algoritmos para adaptar UMC para checar CTLpK. O primeiro elimina o quantificador universal de fórmulas QBF que representam fórmulas CTLpK, e retorna uma fórmula em Forma Normal Conjuntiva – FNC . Os outros dois algoritmos calculam o maior e o menor ponto fixo para as fórmulas modais pertinentes. É mostrado que o conjunto de estados satisfazendo uma fórmula CTLpK qualquer pode

ser caracterizado pelo ponto fixo de uma determinada função. A técnica permite que uma fórmula CTLpK seja traduzida para uma fórmula proposicional $[\alpha](w)$ em FNC, que caracteriza todos os estados do modelo onde α vale.

Especificamente, para uma dada fórmula CTLpK β computa-se a correspondente fórmula proposicional $[\beta](w)$ que codifica os estados do sistema que satisfazem a fórmula. Operacionalmente, o algoritmo trabalha de dentro para fora, desde a subfórmula mais aninhada (os átomos). Para computar $[Oa](w)$, onde O é uma modalidade, trabalha-se assumindo que a fórmula $[a](w)$ já foi computada. Para calcular a tradução utiliza-se ou o ponto fixo ou a caracterização QBF das fórmulas de CTLpK – para fórmulas da forma $O\alpha$ tal que $O \in \{AX, AY, K_i, D_\Gamma, E_\Gamma\}$ utiliza-se o algoritmo de eliminação de quantificador universal de fórmulas QBF, retornando o resultado em Forma Normal Conjuntiva; para fórmulas da forma $O\alpha$ tal que $O \in \{AG, AH, C_\Gamma\}$ usa-se o algoritmo que calcula o maior ponto fixo; para fórmulas da forma $A(\alpha \mathcal{U} \beta)$ utiliza-se o algoritmo que calcula o menor ponto fixo. Desta forma, a partir da fórmula β obtém-se a fórmula proposicional $[\beta](w)$ tal que β é válida no modelo M se e somente se a fórmula proposicional $[\beta](w) \wedge I_\iota(w)$ é satisfatível, isto é, $\iota \in \langle\beta\rangle$.

A tradução apresentada é correta e completa. O principal teorema do artigo atesta que a satisfação de uma fórmula de CTLpK pode ser mapeada em um problema de satisfação de uma fórmula proposicional conjuntiva.

Este é mais um trabalho que apresenta contribuições relevantes sobre como tornar a implementação dos métodos de verificação de propriedades epistêmicas em sistemas não probabilísticos e de tempo discreto factível com o ferramental disponível.

O foco desta tese é o processo de modelagem composicional, especificação e verificação de sistemas multi-agentes no nível do conhecimento. Partimos de um estudo de classes de modelos adequados para métodos de verificação automática, discutimos as atuais abordagens não probabilísticas para obter especificações composicionais para sistemas multi-agentes e investigamos como estender este conceito para modelos envolvendo probabilidade e não determinismo. Discutimos as linguagens presentes na atual literatura que permitem raciocinar sobre conhecimento e construir especificações envolvendo as propriedades epistêmicas e as propriedades sobre a evolução de sistemas pertinentes a cada classe de modelo abordada. Considerando as classes de interesse desta tese, àquelas

para as quais algoritmos para verificação automática de propriedades envolvendo conhecimento não são conhecidos ou ainda não foram estabelecidos, propomos um algoritmo. As principais contribuições apresentadas são o processo de obtenção de modelos globais envolvendo não determinismo e probabilidade a partir de modelos locais; extensão de linguagens de tempo ramificado para possibilitar raciocinar sobre conhecimento e tempo; os respectivos algoritmos para verificação de especificações escritas nas linguagens propostas sobre os modelos propostos; e uma discussão das relações entre incerteza, probabilidade e não determinismo. Apesar de que verificação de propriedades epistêmicas em sistemas não probabilísticos ser um assunto já tratado por outros autores, até o momento a autora desconhece trabalhos semelhantes para modelos probabilísticos.

A vantagem dos processos para verificação automática propostos nesta tese é que a construção do modelo é completamente feita através da composição de (sub)modelos para cada agente, inclusive no que diz respeito a conhecimento – nada além dos modelos para cada agente necessita ser especificado, o modelo global para o sistema multi-agente e as relações epistêmicas sobre as quais se construirá especificações são completamente obtidas no processo de composição do modelo, que pode inclusive ser automatizado. Outro ponto positivo é que os algoritmos apresentados para verificar especificações envolvendo conhecimento sobre os modelos possuem a mesma complexidade que seus equivalentes convencionais, ou seja, não envolvendo conhecimento. Consideramos que estes resultados são um passo relevante para o avanço do conhecimento na área de pesquisa da tese.

A estrutura da tese é a seguinte:

Capítulo 2: O capítulo a seguir introduz e discute lógicas modais para raciocinar sobre propriedades epistêmicas, temporais e probabilísticas. Quatro lógicas modais da literatura atual são abordadas: lógica de conhecimento e crença, lógica modal para conhecimento e tempo, lógica de probabilidades e lógica epistêmica probabilística. O capítulo seguinte estenderá a discussão sobre lógicas modais temporais e probabilísticas adequadas para verificar propriedades de sistemas.

Capítulo 3: Este capítulo apresenta uma revisão dos métodos de verificação automática, sobre modelos convencionais (não probabilísticos) e probabilísticos relevantes para o trabalho desenvolvido nesta tese. Discutiremos o processo geral de verificação

automática de sistemas, e em seguida mostraremos como o método geral é aplicado a três classes distintas de modelos: modelo não probabilístico de tempo discreto (convencional), processos de decisão de Markov (que apresenta não determinismo e escolhas probabilísticas), e cadeias de Markov de tempo contínuo (que modela tempo contínuo, porém não engloba não determinismo). Para cada tipo de modelo é introduzida uma linguagem para escrever especificações sobre o modelo – uma lógica temporal devidamente estendida para expressar propriedades relevantes em cada caso– e também os algoritmos de verificação correspondentes.

Capítulo 4: Onde são apresentadas as principais contribuições desta tese. Três classes de modelos composicionais são apresentados: não probabilístico, probabilístico de tempo discreto, e probabilístico de tempo contínuo. Mostramos como adaptar as linguagens presentes na atual literatura para raciocinar sobre conhecimento e construir especificações envolvendo as propriedades de interesse características de cada classe de modelos abordada. Para cada classe de modelos, apresentamos os respectivos algoritmos para verificação automática de especificações envolvendo conhecimento.

Capítulo 5: O último capítulo faz um resumo dos resultados obtidos, apresenta as conclusões do trabalho de pesquisa e aponta algumas direções para trabalhos futuros.

Capítulo 2

Lógicas modais para raciocinar sobre conhecimento, tempo e probabilidade

Com o advento de sistemas multi-agentes, o interesse em ferramentas e formalismos para raciocinar sobre conhecimento cresceu entre pesquisadores da área de computação, calçado em iniciativas anteriores de diversas áreas interessadas neste tema como por exemplo filosofia [Hin62], economia [Aum76] e inteligência artificial[Moo85].

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. Como um sistema em execução evolui no decorrer do tempo, é esperado que o conhecimento dos agentes envolvidos no sistema também evolua. A interação entre os agentes no decorrer da execução 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 influencia das interações nestes conhecimentos individuais no decorrer do tempo.

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.

O tipo de situação descrito traz atenção para o tratamento de conhecimento e raciocínio em sistemas multi-agentes, assim como a capacidade dos agentes de lidar com incertezas. Apesar de que em algumas situações o grau de incerteza é tal que não permite qualquer estimativa, existem situações onde uma abordagem probabilística pode ser aplicada para modelar incertezas. Em ciência da computação, falamos sobre programas probabilísticos, simulações de comportamento de programas com entradas segundo distribuições probabilísticas, incertezas e aleatoriedade modeladas probabilisticamente.

Uma vez que as incertezas de um agente estejam associadas a fatores probabilísticos, esse fatores devem ser levados em conta nos formalismos utilizados para modelar o conhecimento e raciocínio, bem como as relações entre conhecimento, tempo e probabilidade.

Este capítulo apresenta uma revisão bibliográfica de lógicas modais para raciocinar sobre propriedades epistêmicas, temporais e probabilísticas. Abordaremos quatro lógicas modais da literatura atual: lógica de conhecimento e crença, lógica modal para conhecimento e tempo, lógica de probabilidades e lógica epistêmica probabilística. O capítulo seguinte estenderá a discussão sobre lógicas modais temporais e probabilísticas adequadas para verificar propriedades de sistemas. O estudo de como integrar as noções de conhecimento e suas relações com tempo e probabilidade de forma adequada à verificação automática de propriedades epistêmicas em sistemas multi-agentes é um dos objetivos desta tese.

2.1 Lógica modal de Conhecimento e Crença

Esta seção compreende basicamente uma revisão dos conceitos estabelecidos no trabalho de [FHM95] acerca de conhecimento de um grupo de agentes em um sistema multi-agentes.

2.1.1 Representação do conhecimento de um grupo de agentes

[FHM95] usa 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 – 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 dado por meio de relações entre os mundos possíveis.

Ao considerar que em um grupo de agentes é plausível que cada agente raciocine sobre o conhecimento dos outros agentes, outros conceitos epistêmicos surgem. 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 as muitas possibilidades de raciocínio epistêmico sobre sistemas com múltiplos agentes. 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 [HM84], 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:

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.

Todos tomam conhecimento do fato simultaneamente O fato é anunciado forma que todos os agentes estão presentes ao mesmo tempo, e sabem que todos estão presentes. 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 [PT92], aplicáveis a sistemas assíncronos.

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 num quebra-cabeças. 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.

Consideremos como exemplo um jogo de 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 cada vez menor de possibilidades.

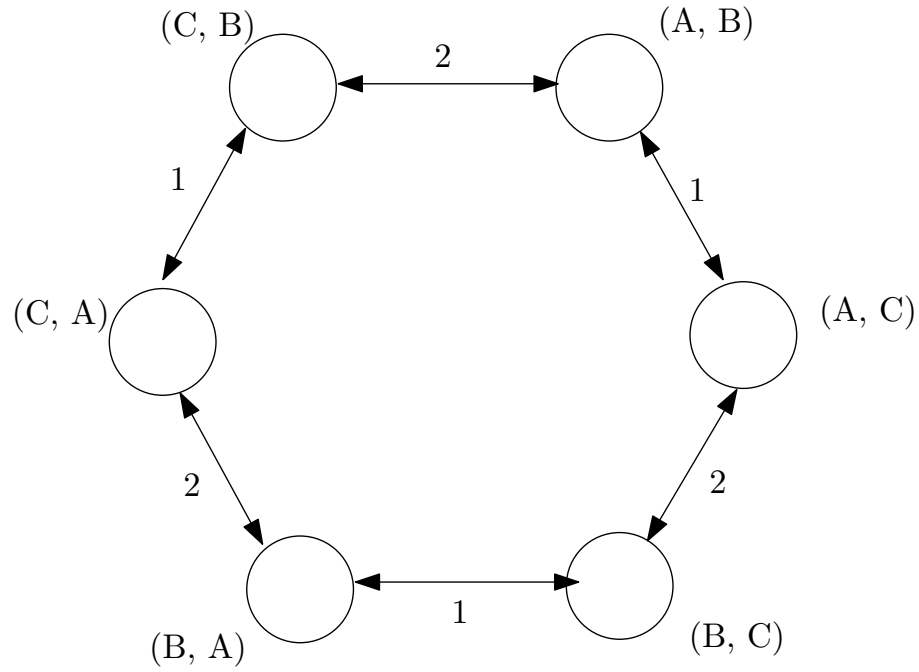


Figura 2.1: Mundos possíveis para os agentes 1 e 2 no jogo das três cartas A, B e C.

Exemplo 2.1.1 *O jogo das três cartas*

Imagine um jogo entre dois agentes, agente $A1$ e agente $A2$, 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 $(x1, x2)$, onde $x1$ representa a carta que está nas mãos do agente $A1$ e $x2$ representa a carta que está nas mãos do agente $A2$. O grafo com os mundos possíveis e as relações de crença dos agentes está representado na figura 2.1 (os ciclos de cada mundo indicando a reflexividade das relações de crença foram omitidos por simplificação).

Considere o raciocínio do agente $A1$ quando ele tem a carta A nas mãos. Nesse caso, ele considera que o agente $A2$ pode ter a carta B ou a carta C. Considera também que se o agente $A2$ tem a carta B então o agente $A2$ pode pensar que o agente $A1$ tem ou a carta A

ou a carta C. E ainda, o agente $A1$ considera que se o agente $A2$ tem a carta B, e que se o agente $A2$ pensa que o agente $A1$ tem a carta C, então o agente $A2$ considera possível que o agente $A1$ pense que o agente $A2$ tem a carta A. Toda essa informação pode ser obtida percorrendo-se um caminho conexo no grafo. Se existe um caminho conexo entre dois estados do grafo, então seguindo-se a linha de raciocínio dos agentes segundo o rótulo das arestas (agentes que crêem - ou consideram possível - 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 grupo formado pelos agentes $A1$ e $A2$ tem 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 $A1$ e o agente $A2$ 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 real distribuição das cartas pode ser inferido no sistema.

2.1.2 Sintaxe e Semântica da lógica modal de conhecimento e crença

Poucos são os sistemas multi-agentes 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". As noções de possibilidade são o objeto principal do estudo da lógica modal. Segundo [HC96] a lógica modal pode ser descrita resumidamente como a lógica da necessidade e da possibilidade, do que "deve ser" e do que "pode ser".

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" ou considera possível. As noções de conhecimento e crença estão fundamentadas nas relações de possibilidade (também chamadas relações de indistinguibilidade ou possibilidade) 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. 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 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.

A linguagem utilizada é a da lógica proposicional modal para um número m de agentes [FHM95]. 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: $\neg, \vee, \wedge, \rightarrow$;
4. Operadores modais: K_i e B_i, $i = 1, 2, 3, \cdots m$ (um para cada agente);

Operadores 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, $K_i\phi$ indica que o agente i sabe ϕ, para $i = 1, 2, 3, \cdots m$;

3. O operador modal B_i indica crença em relação ao agente i, ou seja, $B_i\phi$ indica que o agente i acredita em ϕ, para $i = 1, 2, 3, \cdots m$.

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 $(\neg\phi)$ também é uma fórmula;
3. Se ϕ e ψ são fórmulas, então $(\phi \wedge \psi)$, $(\phi \vee \psi)$ e $(\phi \rightarrow \psi)$ também são fórmulas;
4. Se ϕ é uma fórmula, então $K_i\phi$ e $B_i\phi$, também são fórmulas, para $i = 1, 2, 3, \cdots m$;
5. Nada é uma fórmula, a não ser que seja forçado por um dos ítens acima.

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. Uma estrutura como essa foi proposta por [Kri59].

Definição 2.1.1 *(L_m-frame, também chamado de Estrutura Kripke)*

Um L_m-frame ou Estrutura Kripke $F = (S, \sim_i)$, $i = 1, 2, 3, \cdots m$ é uma estrutura onde:

- *S é um conjunto de estados ou mundos possíveis;*
- *$\sim_i$ é uma relação binária em S, ou seja, um conjunto de pares de elementos de S onde $i = 1, 2, 3, \cdots m$, $(\sim_i \subseteq S \times S)$.*

Definição 2.1.2 *(Modelo Kripke)*

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

Definição 2.1.3 *(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:

$(M,s) \models p$	*se e somente se*	$\pi(s,p) = V$, *onde* $p \in \Phi$;
$(M,s) \models (\neg\phi)$	*se e somente se*	*não é o caso que* $(M,s) \models \phi$;
$(M,s) \models (\phi \wedge \psi)$	*se e somente se*	$(M,s) \models \phi$ *e* $(M,s) \models \psi$;
$(M,s) \models K_i\phi$	*se e somente se*	*para todo* t *tal que* $(s,t) \in \sim_i$ *temos que* $(M,t) \models \phi$;
$(M,s) \models B_i\phi$	*se e somente se*	*existe* t *tal que* $(s,t) \in \sim_i$ *e* $(M,t) \models \phi$;

Seja $M = (F, \pi)$ um modelo para F. Dizemos que M satisfaz ϕ se existe algum mundo $s \in S$ tal que $(M,s) \models \phi$. 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 frame F se é verdadeira em todos os modelos sobre F (para todo M e s, $(M,s) \models \phi$).

2.1.3 Sistemas axiomáticos do conhecimento

Consideremos agora os axiomas e sistemas axiomáticos para conhecimento:

Sistemas axiomáticos do conhecimento

B1 (Axioma Dual) O operador modal B_i pode ser definido em função do operador modal K_i, para $i = 1, 2, 3, \cdots m$, dado o axioma dual: $B_i\phi \leftrightarrow \neg K_i \neg\phi$, para $i = 1, 2, 3, \cdots m$.

Definição 2.1.4 *(Sistema K_m)*

O sistema K_m consiste de dois axiomas e duas regras de inferência:

AXIOMAS	
Tautologias	*Todas as tautologias do cálculo proposicional;*
K_m***1***	$K_i(\phi \rightarrow \psi) \rightarrow (K_i\phi \rightarrow K_i\psi)$ *para todo* ϕ *e* $\psi \in L_m$ *e* $i = 1, 2, \cdots m$.
REGRAS	
Modus Ponens	*De* ϕ *e* $\phi \rightarrow \psi$ *derive* ψ
Generalização do Conhecimento	*De* $\models \phi$ *derive* $K_i\phi$.

Teorema 2.1.1 *O sistema K_m juntamente com o axioma dual é uma axiomatização correta e completa para os mundos de Kripke [MDH85].*

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

A3: Axioma do conhecimento Apenas fatos verdadeiros são conhecidos: $K_i\phi \rightarrow \phi$

A4: Introspecção positiva O agente tem conhecimento de seu conhecimento: $K_i\phi \rightarrow K_i(K_i\phi)$

A5: Introspecção negativa O agente tem conhecimento de sua falta de conhecimento: $\neg K_i\phi \rightarrow K_i\neg K_i\phi$

Teorema 2.1.2 *$A3$, $A4$ 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 2.1.3 *T_m é uma axiomatização correta e completa para os mundos reflexivos [MDH85].*

Teorema 2.1.4 *$S4_m$ é uma axiomatização correta e completa para os mundos reflexivo-transitivos [MDH85].*

Teorema 2.1.5 *$S5_m$ é uma axiomatização correta e completa para os mundos reflexivo-simétrico-transitivos [MDH85].*

2.1.4 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.

A_G: Conhecimento de um agente É o conhecimento relativo a algum agente do grupo sobre um fato ϕ .$(M, s) \models A_G\phi$ se e somente se existe o agente i tal que $i \in G$ e para todo t , se $(s, t) \in R_i$ então $(M, t) \models \phi$;

D_G: Conhecimento Distribuído Ocorre quando unindo o conhecimento individual de todos os agentes do grupo, podemos deduzir ϕ. $(M, s) \models D_G\phi$ se e somente se para todo t se $(s, t) \in \bigcap_{i \in G} R_i$ então $(M, t) \models \phi$;

E_G: Conhecimento de todos O conhecimento de todos de um fato ϕ ocorre quanto todos os agentes do grupo tem conhecimento de ϕ. $(M, s) \models E_G\phi$ se e somente se para todo i tal que $i \in G$, $(M, s) \models K_i\phi$, ou seja: $(M, s) \models E_G\phi$ se e somente se para todo i tal que $i \in G$ então para todo t tal que $(s, t) \in R_i$ então $(M, t) \models \phi$.

C_G: Conhecimento comum 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 ${E_G}^0\phi$ uma representação para ϕ, e seja ${E_G}^{k+1}\phi$ uma representação para $E_G({E_G}^k\phi)$, para $k \geq 1$. Em particular, ${E_G}^1$ é uma representação para $E_G\phi$. $(M, s) \models C_G\phi$ se e somente se $(M, s) \models {E_G}^k$, para $k = 1, 2, 3, \cdots$

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 consequência, acrescentam-se os seguintes axiomas e regras nos sistemas axiomáticos:

Axiomas para conhecimento em grupo de agentes

D1 $K_i\phi \rightarrow D_G\phi$

D2 $D_G\phi \rightarrow D_{G\prime}\phi$, se $G \subseteq G'$

C1 $E_G\phi \rightarrow \bigwedge_{i \in G} K_i\phi$

C2 $C_G\phi \rightarrow E_G(\phi \wedge C_G\phi)$

H1 $C_G\phi \rightarrow {E_G}^k\phi \rightarrow E_G\phi \rightarrow A_G\phi \rightarrow D_G\phi \rightarrow \phi$ *hierarquia do conhecimento*

RC1 De $\models \phi \rightarrow E_G(\phi \wedge \psi)$ derive $\models \phi \rightarrow C_G\psi$ *regra da indução*

Teorema 2.1.6 *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). [FHM95]*

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 este fato é verdadeiro.

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.

2.2 Lógicas temporais e epistêmicas

Muitos sistemas distribuídos são caracterizados pela evolução do conhecimento ao longo de sua execução. A medida que o tempo passa, novas informações são adquiridas pelos agentes e conseqüentemente 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 problemas de consenso.

Uma lógica modal de conhecimento e tempo para sistemas distribuídos com múltiplos agentes foi descrita em [Leh84]. 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 apresentada é capaz de expressar os conceitos de conhecimento de um agente e de um grupo de agentes no decorrer do tempo.

Apresentamos a seguir 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.

2.2.1 Sintaxe e semântica para a Lógica de Conhecimento e Tempo

Chamaremos a lógica para conhecimento e tempo de L_{tm}. A linguagem de L_{tm} é a mesma da lógica de conhecimento e crença L_m para sistemas distribuídos acrescida dos seguintes operadores temporais:

1. Sempre no futuro: $\Box$
2. No próximo passo: $\circ$
3. Até que (*until*): $\mathcal{U}$

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

6. Se ϕ e ψ são fórmulas então $\Box\phi$, $\circ\phi$, $\phi\mathcal{U}\psi$ também são fórmulas.

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

Podemos considerar 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 o 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.

Definição 2.2.1 *(Modelo de Kripke para L_{tm})*

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 $\mathfrak{M} = (S, w, \pi, \sim_1, \sim_2, \cdots, \sim_n)$ é uma estrutura onde:

- *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;*
- *$w \in S^N$ é a história real;*
- *π é uma interpretação (função de valoração) que associa valores verdade às primitivas de Φ em cada estado de S.*
- *Para todos os agentes i, $\sim_i$ é uma seqüencia infinita de relações de equivalência – para todo $k \in N$, $\sim_i^k$ é uma relação de equivalência em S^N.*

O significado intuitivo de $\sim_i^k$ é: duas histórias σ e τ são equivalentes com respeito ao agente i e ao instante k se o conhecimento que i adquiriu sobre o mundo até o instante k não o 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 (ou vice-versa).

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.

Definição 2.2.2 *(Relação R_k)*

Seja $R_k \stackrel{def}{=} (\bigcup_{\forall i} \sim_i^k)^*$ uma relação sobre histórias onde $*$ representa o fecho reflexivo e transitivo da relação.

R_k será usada na definição de satisfação para o conectivo C_G de conhecimento comum.

Definição 2.2.3 *(Satisfatibilidade para L_{tm})*

As fórmulas da linguagem L_{tm} serão avaliadas em de um modelo $\mathfrak{M} = (S, w, \pi, \sim_1, \sim_2, \cdots, \sim_n)$. Para um conjunto de proposições Φ, uma história σ e um instante $k \in N$:

$(\mathfrak{M}, \sigma, k) \models p$	*se e somente se*	$\pi(\sigma, k)(p) = V$, *onde* $p \in \Phi$;
$(\mathfrak{M}, \sigma, k) \models \neg\phi$	*se e somente se*	*não é o caso que* $(\mathfrak{M}, \sigma, k) \models \phi$;
$(\mathfrak{M}, \sigma, k) \models \phi \wedge \psi$	*se e somente se*	$(\mathfrak{M}, \sigma, k) \models \phi$ *e* $(\mathfrak{M}, \sigma, k) \models \psi$;
$(\mathfrak{M}, \sigma, k) \models \circ\phi$	*se e somente se*	$(\mathfrak{M}, \sigma, k+1) \models \phi$
$(\mathfrak{M}, \sigma, k) \models \Box\phi$	*se e somente se*	$(\mathfrak{M}, \sigma, n) \models \phi$ *para todo* $n \geq k$
$(\mathfrak{M}, \sigma, k) \models \phi \mathcal{U} \psi$	*se e somente se*	*para todo* $n \geq k$ *tal que* $(\mathfrak{M}, \sigma, n) \not\models \phi$ *existe algum* m *tal que* $k \leq m \leq n$, *tal que* $(\mathfrak{M}, \sigma, m) \models \psi$
$(\mathfrak{M}, \sigma, k) \models K_i\phi$	*se e somente se*	*para todas as histórias* τ *tais que* $\sigma \sim_i \tau$, $(\mathfrak{M}, \tau, k) \models \phi$
$(\mathfrak{M}, \sigma, k) \models C_G\phi$	*se e somente se*	*para todas as histórias* τ *tais que* $\sigma R_k \tau$, $(\mathfrak{M}, \tau, k) \models \phi$

2.2.2 Sistema aximático para conhecimento e tempo

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. Para L_{tm} adiciona-se a $S5_m$ os seguintes axiomas e regras:

Axiomas

A6 $K_i(\phi \to \psi) \to K_i\phi \to K_i\psi$, para $i = 1, 2 \cdots m$.

A7 $C_G(\phi \rightarrow \psi) \rightarrow C_G\phi \rightarrow C_G\psi$

A8 $C_G\phi \rightarrow \phi$

A9 $C_G\phi \rightarrow K_i C_G\phi$, para $i = 1, 2 \cdots m$.

A10 $C_G(\phi \rightarrow E_G) \rightarrow \phi \rightarrow C_G\phi$

A11 $K_i \circ \phi \rightarrow \circ K_i\phi$, para $i = 1, 2 \cdots m$.

A12 $C_G \circ \phi \rightarrow \circ C_G\phi$.

Regras

R3 De $\models \phi$ derive $\Box\phi$ *generalização do* $\Box$,

R4 De $\models \phi$ derive $C_G\phi$ *generalização do conhecimento comum.*

Teorema 2.2.1 *O sistema $S5_m$ juntamente com os axiomas e regras acima é uma axiomatização correta e completa para L_{tm} [LS82].*

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. Apesar destas restrições a lógica de conhecimento e tempo proposta por [Leh84] 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.

2.3 Lógica de probabilidades

A necessidade de raciocinar sobre probabilidades aparece em muitas áreas de pesquisa. Em ciência da computação, falamos sobre programas probabilísticos, simulações de comportamento de programas com entradas segundo distribuições probabilísticas, incertezas e aleatoriedade modeladas probabilisticamente. Para se trabalhar com raciocínio formal sobre probabilidades faz-se necessária uma lógica para raciocinar sobre probabilidade

com sintaxe e semântica bem definidas. De acordo com [FHM90], muitos autores abordaram lógicas envolvendo probabilidades, porém poucas abordagens permitem raciocinar explicitamente sobre probabilidades.

A lógica apresentada em [FHM90], baseada no trabalho de [Nil86], permite raciocinar sobre probabilidades, não tendo semântica probabilística. Isto significa que todas as suas fórmulas são efetivamente verdadeiras ou falsas, quando avaliadas. O problema da satisfatibilidade para esta lógica é NP-completo.

2.3.1 Sintaxe e Semântica para a Lógica de Probabilidades

Seja $\Phi = \{p_1, p_2, \cdots\}$ um conjunto infinito de proposições ou eventos básicos.

A fórmula $true$ é uma abreviação para $p \vee \neg p$, e a fórmula $false$ uma abreviação para $\neg true$. O conjunto de fórmulas proposicionais ou eventos é o fecho de Φ sobre as operações booleanas $\neg$ e $\wedge$.

Um *termo ponderado primitivo* é uma expressão da forma $w(\phi)$, onde ϕ é uma fórmula proposicional.

Um *termo ponderado*, ou apenas *termo*, é uma expressão da forma $a_1 w(\phi_1) + \cdots + a_k w(\phi_k)$, onde $a_1 \cdots a_k$ são inteiros e $k \geq 1$.

Uma *fórmula ponderada básica* é da forma $t \geq c$, onde t é um termo e c é um inteiro.

Uma *fórmula ponderada* (que também chamaremos de fórmula, por simplicidade) é uma combinação booleana de fórmulas ponderadas básicas. Usaremos as letras f e g para nos referir a fórmulas ponderadas.

Usaremos algumas abreviações para simplificar a notação:

- $w(\phi) - w(\psi) \geq a$ para $w(\phi) + (-1)w(\psi) \geq a$,
- $w(\phi) \geq w(\psi)$ para $w(\phi) - w(\psi) \geq 0$,
- $w(\phi) \leq c$ para $-w(\phi) \geq -c$,
- $w(\phi) < c$ para $\neg(w(\phi) \geq c)$,
- $w(\phi) = c$ para $(w(\phi) \geq c) \wedge (w(\phi) \leq c)$, e
- $w(\phi) \geq c_1/c_2$ para $c_2 w(\phi) \geq c_1$.

A semântica para esta linguagem é dada em função de alguns elementos de teoria de probabilidade.

Definição 2.3.1 *(Espaço de Probabilidades)*

Um espaço de probabilidades é um terno $(S, \mathcal{X}, \mu)$ *onde* S *é um conjunto,* $\mathcal{X}$ *é uma* σ*-algebra de subconjuntos de* S*, cujos elementos são chamados conjuntos mensuráveis, e* μ *é uma medida de probabilidade definida nos conjuntos mensuráveis.* $\mu : \mathcal{X} \rightarrow [0, 1]$ *satisfaz às seguintes propriedades:*

- $\mu(X) \geq 0$*, para todo* $X \in \mathcal{X}$*,*
- $\mu(S) = 1$*,*
- $\mu(\cup_{i=1}^{\infty} X_i) = \sum_{i=1}^{\infty} \mu(X_i)$*, se os conjuntos* X_i *são membros de* $\mathcal{X}$ *disjuntos dois a dois.*

Dado um espaço de probabilidades $(S, \mathcal{X}, \mu)$, a semântica para fórmulas ponderadas pode ser definida associando-se a toda proposição (evento básico) um conjunto mensurável, estendendo essa associação para todos os eventos da forma tradicional, e depois computando a probabilidade destes eventos utilizando μ.

Definição 2.3.2 *(Estrutura de Probabilidades)*

Uma estrutura de probabilidades é uma enupla $\mathfrak{M} = (S, \mathcal{X}, \mu, \pi)$*, onde* $(S, \mathcal{X}, \mu)$ *é um espaço de probabilidades, e* π *uma função de valoração que associa a cada estado de* S *um valor verdade às proposições primitivas de* Φ*,* $\pi(s)(p) \in \{true, false\}$ *para todo* $s \in S$ *e* $p \in \Phi$*. Definiremos também* $p^{\mathfrak{M}} = \{s \in S | \pi(s)(p) = true\}$.

Dizemos que uma estrutura de probabilidades $\mathfrak{M}$ é mensurável se para cada proposição primitiva p, o conjunto $p^{\mathfrak{M}}$ é mensurável.

A primeira abordagem para a semântica da lógica de probabilidades se concentra em estruturas de probabilidade mensuráveis. O conjunto $p^{\mathfrak{M}}$ pode ser pensado como o conjunto de mundos possíveis onde p é verdadeiro, ou o conjunto de estados onde o evento p ocorre. Estenderemos $\pi(s)$ para todas as fórmulas proposicionais da maneira usual, e então associaremos com cada fórmula proposicional ϕ o conjunto $\phi^{\mathfrak{M}} = \{s \in S | \pi(s)(\phi) = true\}$. $\phi^{\mathfrak{M}}$ é mensurável.

Definição 2.3.3 *(Satisfação para fórmulas ponderadas para o caso mensurável)*

Seja $\mathfrak{M} = (S, \mathcal{X}, \mu, \pi)$.

$\mathfrak{M} \models a_1 w(\phi_1) + \cdots + a_k w(\phi_k) \geq c$ *se e somente se* $a_1 \mu(\phi_1^{\mathfrak{M}}) + \cdots + a_k \mu(\phi_k^{\mathfrak{M}}) \geq c$

A satisfação para fórmulas ponderadas arbitrárias é obtida da forma usual:

$\mathfrak{M} \models \neg f$ *se e somente se* $\mathfrak{M} \not\models f$

$\mathfrak{M} \models f \wedge g$ *se e somente se* $\mathfrak{M} \models f$ *e* $\mathfrak{M} \models g$

Dizemos que uma fórmula ponderada f é valida se $\mathfrak{M} \models f$ para todas as estruturas de probabilidade $\mathfrak{M}$, e que f é satisfatível se $\mathfrak{M} \models f$ para alguma estrutura de probabilidade $\mathfrak{M}$.

2.3.2 Sistemas axiomáticos para lógica de probabilidades

Apresentamos a seguir o sistema axiomático para o caso de semântica mensurável da lógica de probabilidade, que chamaremos AX_{mens}. O sistema se divide naturalmente em três partes, que tratam respectivamente de raciocínio proposicional, raciocínio sobre desigualdades lineares, e raciocínio sobre probabilidades.

1. Raciocínio proposicional

   Tautologias Todas as instâncias das tautologias proposicionais.

   Modus Ponens De f e $f \Rightarrow g$ infira g

2. Raciocínio sobre desigualdades lineares

   In1 $(a_1 w(\phi_1) + \cdots + a_k w(\phi_k)) \geq c \Leftrightarrow (a_1 w(\phi_1) + \cdots + a_k w(\phi_k) + 0 w(\phi_{k+1})) \geq c$ *(soma e subtração de termos nulos)*

   In2 $(a_1 w(\phi_1) + \cdots + a_k w(\phi_k)) \geq c \Rightarrow (a_{j_1} w(\phi_{j_1}) + \cdots + a_{j_k} w(\phi_{j_k}) \geq c)$, se $j_1 \cdots j_k$ é uma permutação de $1 \cdots k$ *(permutação)*

   In3 $(a_1 w(\phi_1) + \cdots + a_k w(\phi_k)) \geq c \wedge (a'_1 w(\phi_1) + \cdots + a'_k w(\phi_k)) \geq c' \Rightarrow (a_1 + a'_1) w(\phi_1) + \cdots + (a_k + a'_k) w(\phi_k) \geq (c + c')$ *(adição de coeficientes)*

   In4 $(a_1 w(\phi_1) + \cdots + a_k w(\phi_k)) \geq c \Leftrightarrow (d a_1 w(\phi_1) + \cdots + d a_k w(\phi_k)) \geq dc$ se $d > 0$ *(Multiplicação e divisão de coeficientes não nulos).*

In5 $t \geq c \vee t \leq c$, se t é um termo *(dicotomia)*

In6 $t \geq c \Rightarrow (t > d)$ se t é um termo e $c > d$ *(monotonicidade)*

3. Raciocínio sobre probabilidades

w1 $w(\phi) \geq 0$

w2 $w(true) = 1$

w3 $w(\phi \wedge \psi) + w(\phi \wedge \neg\psi) = w(\phi)$ (adição)

w4 $w(\phi) = w(\psi)$ se $\phi \equiv \psi$ é uma tautologia proposicional (distributividade).

Teorema 2.3.1 *AX_{MEAS} é uma axiomatização correta e completa com respeito a estruturas de probabilidade mensuráveis [FHM90].*

Quando não é o caso que os conjuntos $\phi^{\mathfrak{M}}$ associados a cada evento ϕ são mensuráveis, ou quando não se deseja associar probabilidades a todos os eventos, ainda assim podemos raciocinar utilizando limites inferiores e superiores para a probabilidade, e modificando a semântica da lógica de probabilidades adequadamente.

Uma semântica natural para o caso de $\phi^{\mathfrak{M}}$ não mensurável pode ser obtida considerando o limite inferior induzido pela distribuição de probabilidades.

Definição 2.3.4 *(Limite de probabilidade inferior)*

Dado um espaço de probabilidades $(S, \mathcal{X}, \mu)$ e um subconjunto arbitrário A de S, o limite de probabilidade inferior para A induzido por μ é definido como $\mu_*(A) = sup\{\mu(B)|B \subseteq A$ e $B \in \mathcal{X}\}$.

μ_* é definido para todos os subconjuntos de S, e $\mu_*(A) = \mu(A)$ se A é mensurável.

Definição 2.3.5 *(Satisfação para fórmulas ponderadas)*

Seja $\mathfrak{M} = (S, \mathcal{X}, \mu, \pi)$.

$\mathfrak{M} \models a_1 w(\phi_1) + \cdots a_k w(\phi_k) \geq c$ se e somente se $a_1 \mu_*(\phi_1^{\mathfrak{M}}) + \cdots + a_k \mu_*(\phi_k^{\mathfrak{M}}) \geq c$

A satisfação para fórmulas ponderadas arbitrárias é obtida da mesma forma utilizada no caso mensurável.

A axiomatização também sofre alterações. Primeiro, o axioma w3 não vale para o caso geral, e é retirado.

O axioma w5 atestando que $w(false) = 0$ tem que ser incluído.

w5 $w(false) = 0$

Um último axioma deve ser acrescentado para tornar o sistema axiomático correto e completo para a lógica probabilística no caso geral, porém para definir o axioma w6 mais algumas definições são necessárias.

Definição 2.3.6 *(n-átomo)*

Um n-átomo é uma fórmula da forma $p'_1 \wedge \cdots \wedge p'_n$, onde p'_i é p_i ou $\neg p_i$, para cada i.

Se n é subentendido ou não importante, usaremos a forma simplificada átomo para nos referirmos a n-átomos.

Definição 2.3.7 *(n-região)*

Seja $\sigma_1, \cdots, \sigma_{2^n}$ uma lista de todos os n-átomos, em uma ordem fixa. Definimos uma n-região como uma disjunção de n-átomos, onde os n-átomos aparecem em ordem nos termos que compõe a disjunção.

Desta forma, garantidamente há 2^{2^n} n-regiões distintas, uma correspondente a cada subconjunto dos n-átomos.

Identificaremos a disjunção vazia como a fórmula $false$. Toda a fórmula proposicional em que todas as proposições primitivas estão em $\{p_1, \cdots, p_n\}$ é equivalente a uma n-região.

Definição 2.3.8 *(Região de tamanho r)*

Uma região de tamanho r é uma região que contém exatamente r termos disjuntos.

Dizemos que ρ' é uma subregião de ρ se ρ e ρ' são n-regiões e cada termo disjunto de ρ' é um termo disjunto de ρ. Neste caso, $\rho' \Rightarrow \rho$ é uma tautologia.

Uma subregião de tamanho r de uma região ρ é uma região de tamanho r que é subregião de ρ.

Podemos agora enunciar o axioma w6:

w6 $\sum_{t=1}^{r} \sum_{\rho' \text{uma subregião de tamanho t de } \rho} (-1)^{r-t} w(\rho') \geq 0$,
se ρ é uma região de tamanho r e $r \geq 1$.

O sistema axiomático inclui um axioma w6 para cada n, cada n-região ρ, e cada r para $1 \leq r \leq 2^{2^n}$.

Teorema 2.3.2 *O problema de decidir se uma fórmula ponderada é satisfatível com respeito a estruturas genéricas de probabilidade é NP-completo.*

2.4 Lógica epistêmica probabilística

Em muitas das áreas em que raciocinar sobre conhecimento se mostra útil, é importante ser capaz de raciocinar também sobre a probabilidade de certos eventos assim como o conhecimento de agentes. Exemplos de aplicação para raciocínio probabilístico e epistêmico são sistemas distribuídos envolvendo programas randômicos ou probabilísticos. Conhecimento e probabilidade já foram anteriormente considerados conjuntamente, como por exemplo em artigos sobre economia [Aum76], onde as probabilidades são embutidas no modelo. A lógica proposta em [FHM90], descrita na seção 2.3, foi consolidada e estendida em [FH94] de forma a permitir raciocinar sobre conhecimento e probabilidades. Tal lógica epistêmica probabilística é descrita a seguir.

2.4.1 Sintaxe e Semântica da Lógica Epistêmica Probabilística

A linguagem estende aquela mostrada na seção 2.3 [FHM90] de duas formas. Ao invés de termos um único operador modal probabilístico w, teremos um operador w_i para cada agente i do sistema, para poder raciocinar sobre a probabilidade associada a cada evento por diferentes agentes. E as fórmulas que aparecem no escopo de um operador modal probabilístico podem ser arbitrárias, ao invés de meramente proposicionais. Em particular, fórmulas de ordem superior como por exemplo $w_i(w_j(\phi) \geq b) \geq c$ são permitidas. Intuitivamente, $w_i(\phi) \geq b$ expressa que "de acordo com o agente i, fórmula ϕ vale com probabilidade pelo menos b".

Se $\phi_1, \cdots, \phi_k$ são fórmulas, então $a_1 w_i(\phi_1) + \cdots + a_k w_i(\phi_k) \geq b$ onde $a_1, \cdots, a_k, b$ são números racionais arbitrários e $k \geq 1$, também é uma fórmula. Chamamos este tipo de fórmula de "fórmula de i-probabilidade", ou simplesmente de "fórmula de probabilidade"

se não desejamos especificar i. Chamamos de termo uma expressão da forma $a_1 w_i(\phi_1) + \cdots + a_k w_i(\phi_k)$.

Novamente, faremos uso de abreviações para simplificar a notação:

- $w_i(\phi) - w_i(\psi) \geq a$ para $w(\phi) + (-1)w(\psi) \geq a$,
- $w_i(\phi) \geq w_i(\psi)$ para $w_i(\phi) - w_i(\psi) \geq 0$,
- $w_i(\phi) \leq b$ para $-w_i(\phi) \geq -b$,
- $w_i(\phi) < b$ para $\neg w_i(\phi) \geq b$,
- $w_i(\phi) = b$ para $(w_i(\phi) \geq b) \wedge (w_i(\phi) \leq b)$, e
- $w_i(\phi) \geq b_1/b_2$ para $b_2 w_i(\phi) \geq b_1$.

Também usamos $K_i^b(\phi)$ como uma abreviação para $K_i(w_i(\phi) \geq b)$. Intuitivamente, essa fórmula significa que "o agente i sabe que a probabilidade de ϕ é maior ou igual a b". Pode parecer que a fórmula $w_i(\phi) \geq b$ já deveria por si só ter este mesmo significado, mesmo sem o operador K_i, mas este não é o caso na semântica de [FH94] – em um dado estado s, a modalidade $w_i(\phi)$ representa a probabilidade de ϕ de acordo com a distribuição de probabilidades atribuída ao agente i no estado s. Este é um recurso útil para modelar o desconhecimento que o agente i pode ter em relação a que distribuição probabilística está sendo usada para calcular $w_i\phi$. Por exemplo, se o agente i sabe que uma entre duas distribuições governa ϕ, e de acordo com uma a probabilidade de ϕ é $1/2$, e de acordo com outra a probabilidade de ϕ é $3/4$, isto pode ser modelado de forma que haja dois estados que i não pode distinguir, tal que em um deles $w_i(\phi) = 1/2$ e no outro $w_i(\phi) = 3/4$. Nesta situação é o caso que $K_i(w_i(\phi) \geq 1/2)$ vale.

Conforme já observado nas seções 2.1 e 2.2, os modelos padrão para semânticas de lógicas epistêmicas são os Modelos de Kripke [Kri59]– modelos com relações de possibilidade para todos os agentes envolvidos. As relações de possibilidade são interpretadas de um ponto de vista epistêmico: um mundo é acessível para um agente se e somente se é consistente com a informação que o agente possui. Para adicionar probabilidade a estes modelos, Fagin e Halpern definem modelos epistêmicos probabilísticos, baseados

em estruturas Kripke, onde um espaço de probabilidades é associado a cada agente em cada mundo possível. O primeiro passo é a construção da estrutura Kripke.

Definição 2.4.1 *(Estrutura Kripke para Conhecimento Probabilístico)*

Considerando um sistema com $1, \cdots, n$ *agentes e um conjunto* Φ *de proposições atômicas, uma Estrutura Kripke para Conhecimento e Probabilidade (para* n *agentes) é uma tupla* $(S, \pi, \sim_1, \cdots, \sim_n, \mathcal{P})$*, onde* S *é um conjunto de estados,* $\pi(s)$ *é uma atribuição de valores verdade às proposições primitivas de* Φ *em cada estado* $s \in S$*,* $\sim_i$ *é uma relação de equivalência sobre os estados de* S*, para* $i = 1, \cdots, n$*.* $\mathcal{P}$ *é uma atribuição de probabilidades, que atribui a cada agente* $i \in \{1, \cdots, n\}$ *e estado* $s \in S$ *um espaço de probabilidades* $\mathcal{P}(i, s) = (S_{i,s}, \mathcal{X}_{i,s}, \mu_{i,s})$ *, onde* $S_{i,s} \subseteq S$*.*

Usaremos $\mathcal{P}_{i,s}$ como uma abreviação para $\mathcal{P}(i, s)$. Intuitivamente, o espaço de probabilidades $\mathcal{P}_{i,s}$ descreve as probabilidades que o agente i associa a eventos, dado que o estado é s.

A intuição da relação $\sim_i$ é mesma apresentada na seção 2.1.

A semântica das fórmulas que não envolvem probabilidade é dada da mesma forma que foi apresentado na seção 2.1. Para dar semântica a fórmulas envolvendo probabilidade, assumimos indutivamente que tenhamos definido $(M, s) \models \phi$ para cada estado $s \in S$. Seja $S_{i,s}(\phi) = \{s' \in S_{i,s} | (M, s') \models \phi\}$. A semântica de fórmulas da forma $w_i(\phi) \geq b$ é dada da seguinte forma:

$$(M, s) \models w_i(\phi) \geq b \quad \text{se e somente se} \quad (\mu_{i,s})_*(S_{i,s}(\phi)) \geq b.$$

O uso do limite inferior de probabilidade justifica-se para contemplar os casos onde o conjunto $S_{i,s}(\phi)$ não é mensurável.

De forma geral, temos:

$$(M, s) \models \quad a_1 w_i(\phi_1) + \cdots + a_k w_i(\phi_k) \geq b \text{ se e somente se}$$
$$a_1(\mu_{i,s})_*(S_{i,s}(\phi_1)) + \cdots + a_k(\mu_{i,s})_*(S_{i,s}(\phi_k)) \geq b.$$

2.4.2 Sistema axiomático para conhecimento probabilístico

A axiomatização natural para a lógica epistêmica probabilística combina axiomas da lógica epistêmica com axiomas da lógica de probabilidades, adequadamente adaptados aos operadores modais probabilísticos para conhecimento.

Os axiomas podem ser divididos em quatro componentes – a primeira para raciocínio proposicional, a segunda para raciocínio sobre conhecimento, a terceira para raciocínio sobre desigualdades e a quarta, para raciocínio sobre probabilidades.

As componentes para raciocínio proposicional e raciocínio sobre desigualdades são as mesmas apresentadas para as respectivas componentes axiomáticas da lógica de probabilidades na seção 2.3.2.

A componente para raciocínio sobre conhecimento corresponde aos axiomas do sistema $S5_m$, apresentado em 2.1.3.

A componente para raciocínio sobre probabilidades é diretamente adaptada dos respectivos axiomas da lógica de probabilidades (seção 2.3), com a inclusão de w5.

w1 $w_i(\phi) \geq 0$

w2 $w_i(true) = 1$

w3 $w_i(\phi \wedge \psi) + w_i(\phi \wedge \neg\psi) = w_i(\phi)$ (adição)

w4 $w_i(\phi) = w_i(\psi)$ se $\phi \equiv \psi$ é uma tautologia proposicional (distributividade).

w5 $w_i(\text{falso}) = 0$.

O axioma w5 é redundante, porém será importante ao substituir w3 por outro axioma que se aplique ao caso não mensurável, em que w3 não é correto. Podemos substituí-lo por outro axioma para obter uma axiomatização completa:

w6 $w_i(\phi_1 \vee \cdots \vee \phi_k) \geq \sum_{\mathcal{I} \subseteq \{1,\cdots,k\}, \mathcal{I} \neq 0} (-1)^{|\mathcal{I}|+1} w_i(\bigwedge_{i \in \mathcal{I}} \phi_i)$.

Seja AX_{MEAS} composto pelos axiomas e regras das componentes para raciocínio proposicional, raciocínio sobre conhecimento, raciocínio sobre desigualdades e os axiomas w1 a w5 para raciocínio sobre probabilidades. Seja AX obtido a partir de AX_{MEAS} substituindo w3 por w6.

Teorema 2.4.1 *AX (respectivamente AX_{MEAS}) é uma axiomatização correta e completa para a lógica epistêmica probabilística (respectivamente, para estruturas satisfazendo a restrição de que para todo i, todo s e toda fórmula ϕ, o conjunto $S_{i,s}(\phi) \in \mathcal{X}_{i,s}$).*

Capítulo 3

Modelagem e Verificação de Sistemas

A verificação formal através da checagem de modelos, conhecida como "Model Checking", é uma técnica poderosa e madura de verificação de sistemas [CE81]. A crescente relevância de sistemas multi-agentes fez com que os métodos de modelagem e verificação formal por meios computacionais fossem ainda mais valorizados.

De uma forma geral, os métodos de verificação formal analisam os possíveis comportamentos dos sistemas, de forma exaustiva. Existem métodos baseados em simulação, teste ou raciocínio dedutivo, além do método que abordamos neste trabalho – chamado de método de verificação de modelos ou método de checagem de modelos (*Model Check* - MC).

Comparado com as outras abordagens, o método da checagem de modelos tem duas grandes vantagens [PCG00]:

- Sua aplicação é completamente automática;
- Produz um contra-exemplo quando o modelo não satisfaz a propriedade desejada, mostrando o comportamento do sistema que falha em satisfazer a propriedade desejada.

O método consiste na verificação de um modelo do sistema desejado, através de enumeração exaustiva, implícita ou explícita, dos estados alcançáveis pelo sistema e os comportamentos que emergem destes estados. As especificações são expressas em lógicas temporais, e os sistemas são modelados como sistemas de transição de estados. Um procedimento eficiente de busca é utilizado para determinar se a especificação é verdadeira

no modelo. Especificações parciais podem ser utilizadas, permitindo a obtenção de informações relevantes sem a necessidade de uma especificação completa.

A maior desvantagem do método é o problema conhecido como "problema da explosão de estados", que pode ocorrer se o sistema tem muitas componentes realizando transições em paralelo. Neste caso o número de estados globais do sistema pode crescer exponencialmente com o numero de processos. Porém progressos foram feitos e ao final dos anos 80 o tamanho dos sistemas de transição aos quais o método pode ser aplicado com eficiência aumentou bastante, com o advento do método de checagem de modelos simbólica (*simbolic model check*), técnica que emprega diagramas de decisão binários ordenados (*ordered binary decision diagrams*, OBDDs), uma estrutura de dados para representar funções booleanas. As novas estruturas tornaram possível obter representações concisas e manipulá-las rapidamente.

[RKNP04] apresenta uma revisão de como empregar métodos de verificação automática sobre modelos probabilísticos. Através do uso de probabilidades é possível modelar comportamentos não confiáveis ou imprevisíveis, como sistemas tolerantes a falhas, protocolos capazes de lidar com canais de comunicação com perda de mensagens e estimativas de tempo de espera por recursos, entre outras coisas. Em comum com o processo convencional de verificação de modelos, a verificação probabilística de modelos envolve a análise de estados alcançáveis em um sistema de transição de estados, porém incorpora também o cálculo de estimativas através de métodos numéricos ou analíticos. Enquanto processos de checagem de modelos convencionais trabalham com modelos de estados e transições e especificações feitas em lógicas temporais, na verificação probabilística os modelos considerados são variantes de cadeias de Markov, onde além de estados e transições estão também codificadas as probabilidades de que uma transição seja feita.

Apesar de que os algoritmos básicos para verificação de modelos probabilísticos foram desenvolvidos em 1980, os trabalhos de implementação, ferramentas e aplicações são em sua maioria recentes, especialmente as que permitem especificações em lógicas temporais probabilísticas [RKNP04]. Muitos investimentos científicos e de implementação têm sido feitos nesta área nos últimos anos.

Há muitas aplicações para probabilidades na modelagem e análise de sistemas. Neste trabalho estamos especialmente interessados em sistemas concorrentes. Muitos aspectos

de sistemas concorrentes podem ser modelados probabilisticamente, enriquecendo os processos de análise e verificação formal de sistemas. Um resultado conhecido de [FLP85] é que não há soluções simétricas para certos problemas de sistemas concorrentes na presença de falhas, como por exemplo consenso distribuído entre processos na presença de falhas. Porém, ao enfraquecer a propriedade "o algoritmo eventualmente termina" para "o algoritmo termina com probabilidade 1", e usando escolhas randômicas, é garantido que uma solução existe [Rab82], [AH90]. Existem vários sistemas concorrentes para os quais uso de abordagens randômicas provê soluções simples, elegantes e eficientes. Em outros casos, um modelo envolvendo não determinismo e escolhas probabilísticas é essencial. Vale ressaltar que ao mencionar não determinismo estamos nos referindo a eventos não determinísticos que não possuem probabilidades associadas, ou seja, cuja probabilidade não se pode estimar (não determinismo puro). Nesta tese, adotaremos o termo não determinismo para nos referir a não determinismo puro.

Este capítulo apresenta uma revisão bibliográfica dos métodos de verificação que são relevantes para o trabalho desenvolvido nesta tese. Discutiremos o processo geral de verificação automática de sistemas, e em seguida mostraremos como o método geral é instanciado para trabalhar com três tipos de modelos: modelo não probabilístico de tempo discreto (convencional), processos de decisão de Markov (que apresenta não determinismo e escolhas probabilísticas), e cadeias de Markov de tempo contínuo (que modela tempo contínuo, porém não engloba não determinismo). Para cada tipo de modelo é introduzida uma linguagem para escrever especificações sobre o modelo – uma lógica temporal devidamente estendida para expressar propriedades relevantes em cada caso– e também os algoritmos de verificação correspondentes.

3.1 Modelagem e verificação de sistemas não probabilísticos

Esta seção apresenta uma revisão dos conceitos e algoritmos básicos envolvidos no método de checagem explícita de modelos não probabilísticos. O método de verificação explícita é o mais intuitivo dos métodos de verificação automática, e por isso foi escolhido para base dos estudos realizados nesta tese. Esta revisão foi baseada em [PCG00]. O processo de checagem de modelos pode ser dividido em três etapas principais: modelagem do sistema, especificação das propriedades a serem verificadas, e finalmente a verificação ou checagem do modelo. Esta mesma divisão é usada a seguir para apresentar o processo completo.

3.1.1 Modelagem de Sistemas

Ao pensar em verificar propriedades de um sistema, é necessário pensar em que tipo de propriedades se deseja verificar. Esta noção é importante na hora de modelar um sistema, pois o nível de detalhamento do modelo depende da especificidade das propriedades que se deseja investigar. Para que o modelo seja adequado ao processo de verificação formal, ele deve ser capaz de capturar as propriedades desejadas. Além disso, o modelo deve também permitir a abstração de detalhes que não interferem nas propriedades a serem verificadas, evitando que este fique carregado de informações que aumentam a complexidade do processo e não contribuem com o resultado almejado. Por exemplo, ao se pensar em um protocolo de comunicação, podemos focar na troca de mensagens e ignorar o conteúdo das mensagens em questão.

Os sistemas de interesse do trabalho desta tese são os sistemas concorrentes, também chamados de sistemas distribuídos ou multi-agentes. Esta classe de sistemas recai naturalmente na categoria de *sistemas reativos*. Um sistema reativo é aquele que muda seu comportamento em resposta a estímulos. É um sistema orientado a eventos que reage continuamente a estímulos internos ou externos. [PCG00]

Autômatos finitos são uma forma natural de descrever comportamentos dinâmicos de sistemas reativos. Autômatos finitos são formalmente definidos e facilmente computáveis por programas de computador. Como um modelo de transição de estado, um autômato

finito possui dois elementos básicos: estados e transições. Ao utilizar o formalismo de Autômatos Finitos para modelar sistemas concorrentes, fazemos um mapeamento natural entre conceitos abstratos de um sistema e elementos do formalismo:

Estados Um estado é uma "fotografia" ou descrição instantânea do sistema. Uma forma simples de representar o estado é através de um vetor com o valor de todas as variáveis envolvidas.

Transições As transições representam as mudanças de estado no sistema. As transições entre os estados são disparadas por ações ou eventos internos ou externos ao sistema, e ligam, no autômato finito que modela o sistema, o estado que antecede ao que sucede a ação ou evento em questão.

Para modelar sistemas reativos através de autômatos finitos, devemos incorporar no modelo as propriedades relevantes do sistema a cada estado. Para isso, enriquecemos o modelo baseado em autômato para uma *estrutura Kripke*. Uma estrutura Kripke possui um conjunto de estados, um conjunto transições entre os estados, e também uma função que rotula cada estado com um conjunto de propriedades que são válidas neste estado.

Definição 3.1.1 *(Estrutura Kripke)*

$\mathfrak{S} = \langle S, S_0, R \rangle$ *é dito uma estrutura Kripke se satisfaz as seguintes propriedades:*

- S *é um conjunto não vazio de estados,*
- $S_0 \subseteq S$ *e* $S_0 \neq \emptyset$*, sendo* S_0 *o conjunto de estados iniciais do sistema.*
- $R \subseteq S \times S$ *é uma relação de transição, necessariamente uma função total, de forma que todo estado em* S *tenha necessariamente um sucessor através da relação de acessabilidade* R*.*

Definição 3.1.2 *(Modelo Kripke)*

Seja $\mathcal{P}$ *um conjunto de proposições atômicas, e* $\mathfrak{S} = \langle S, S_0, R, \mathcal{P} \rangle$ *uma estrutura Kripke.* $\mathfrak{M} = \langle S, S_0, R, \mathcal{P}, \mathcal{L} \rangle$ *é um modelo Kripke para* $\mathfrak{S}$*, onde:*

- $\mathcal{L} : S \rightarrow 2^{\mathcal{P}}$ *é a função que rotula cada estado com o conjunto de proposições atômicas que são verdadeiras naquele estado.*

Uma vez definido o modelo geral para sistemas reativos, voltaremos nossa atenção para sistemas multi-agentes, também chamados de sistemas concorrentes. Um sistema multi-agente envolve uma série de componentes, chamadas de agentes, que executam em conjunto. Normalmente as componentes têm meios para comunicarem-se, o que torna a computação de um sistema multi-agente mais interessante do que o simples paralelismo de processos. O comportamento de um sistema multi-agente tende a ser complexo, dado que o comportamento é ditado por interações e colaborações advindas de diferentes componentes autônomas – os agentes. Para falar sobre eles e capturar suas propriedades, o modelo adotado deve ser leve e flexível, capaz de capturar as interações e informações de cada agente e também o comportamento conjunto resultante de todos os agentes executando ao mesmo tempo, o que dita o comportamento global do sistema.

Num sistema multi-agente cada agente possui seu próprio comportamento (agentes autônomos), ditado por um programa local. O comportamento (ou *computação*) de um sistema multi-agentes é ditado pelas interações dos programas locais que cada agente executa. Este é um conceito fundamental para sistemas multi-agente, e justo o que o torna tão interessante.

[FHM95] apresenta um modelo baseado em autômatos para sistemas multi-agentes, conhecido como modelo de Sistemas Interpretados. Este modelo é atualmente uma referência na área de sistemas multi-agentes. Neste modelo, um sistema é visto como um conjunto de execuções. A definição de execução é construída sobre outras definições elementares para agentes, estado local, estado global e estado do ambiente.

Definição 3.1.3 *(Execução)*

Sejam:

- $Ag = \{1, \cdots, n\}$ *um conjunto de agentes;*
- L_i *o conjunto de possíveis estados locais de um agente* $i \in Ag$*;*
- L_e *o conjunto de possíveis estados do ambiente;*
- $G \subseteq L_1 \times \cdots \times L_n \times L_e$ *o conjunto de estados globais do sistema.*

Uma "execução" é uma função $r : \mathbb{N} \rightarrow G$ *que associa a cada número natural* $u \in \mathbb{N}$ *um estado global* $r(u)$*.*

Cada cada estado local de L_i contém toda informação necessária para caracterizar completamente o estado em questão – por exemplo, se os agentes considerados são agentes de software, o estado local seria composto pelo valor das variáveis locais e o valor do contador de programa (*program counter*).

A informação disponível para um agente num determinado momento é determinada pelo seu estado local. O estado do sistema em um ponto no tempo é caracterizado por uma tupla $\langle l_1, \cdots, l_n \rangle$, onde $l_i \in L_i$ é o estado local do agente i neste momento.

Em geral, o sistema possui um conjunto de execuções (computações) possíveis, o que permite definir "sistema" como um conjunto de execuções. Uma execução junto com um instante no tempo representa um "ponto" – um ponto (r, u) define um estado global $r(u)$. A i-ésima componente de uma tupla $r(u)$ é denotada por $r_i(u)$, que representa o estado local do agente i na execução r e no tempo u.

Definição 3.1.4 *(Sistema) Um sistema é um conjunto $\mathcal{R}$ não vazio de execuções sobre um mesmo contradomínio G.*

Definição 3.1.5 *(Sistema Interpretado)*

Um "sistema interpretado" $\mathcal{I}$ sobre um conjunto Φ de proposições atômicas é um par $\langle \mathcal{R}, \pi \rangle$, onde:

- *$\mathcal{R}$ é um Sistema, e*
- *$\pi : \mathcal{R} \times \mathbb{N} \rightarrow 2^{\Phi}$ é a função de valoração, que determina o conjunto de proposições primitivas verdadeiras a cada ponto de $\mathcal{R}$.*

3.1.2 Especificação: Lógica de computação ramificada CTL

A lógica utilizada nas especificações deve ser capaz de expressar diretamente as propriedades de interesse ao lidar com sistemas concorrentes. Lógicas temporais são um formalismo para descrever seqüências de transições entre estados em um sistema reativo. Uma lógica temporal com os operadores modais de tempo básicos foi apresentada na seção 2.2. Nesta seção, focaremos na lógica de computação ramificada (*Computation Tree Logic* – CTL).

Em [BAMP81], Ben-Ari et al. apresentaram pela primeira vez a lógica de computação ramificada, com o objetivo de lidar com o conjunto de execuções possíveis (árvore de execução) de um dado programa. Esta lógica foi especialmente projetada para lidar com as conseqüencias trazidas por não determinismo, bem como as geradas por programas que interagem assincronamente. Foi em [CE81] que Emerson e Clarke deram a forma final a CTL, provendo um procedimento de decisão.

Conceitualmente, fórmulas em CTL descrevem propriedades de árvores de computação. As árvores de computação são obtidas considerando uma estrutura Kripke com um estado inicial determinado, e expandindo os estados e arestas da estrutura Kripke de acordo com as possíveis execuções a partir do estado inicial.

Sintaxe e Semântica para CTL

As fórmulas de CTL são formadas por operadores temporais imediatamente precedidos por quantificadores de caminhos, a saber:

- $\exists$X (na próxima unidade de tempo), requer que exista um caminho para o qual uma propriedade seja válida no segundo estado do caminho.
- $\exists$F (eventualmente), requer que exista um caminho para o qual uma propriedade seja válida em algum estado do caminho.
- $\exists$G (sempre ou globalmente), requer que exista um caminho no qual uma propriedade valha em todos os estados do caminho.
- $\exists$U (until), operador que relaciona duas propriedades da seguinte forma: requer que exista um caminho no qual em um estado no caminho a segunda propriedade seja válida, e que em todos os estados que antecedam a este no caminho, a primeira propriedade seja válida.

Seja $\mathcal{P}$ um conjunto de proposições. A linguagem das fórmulas de CTL é definida como a seguir:

$ForCTL(\mathcal{P})$é o menor conjunto For de fórmulas tal que:

- $p \in For$ se e somente se $p \in \mathcal{P}$,

- $\{\neg\phi_1, \phi_1 \vee \phi_2, \exists \mathsf{X}\phi_1, \exists \mathsf{G}\phi, \exists[\phi_1 \mathsf{U} \phi_2]\} \subseteq For$ se e somente se $\{\phi_1, \phi_2\} \subseteq For$.

O restante dos operadores proposicionais é definido em termos de negação ("$\neg$") e disjunção ("$\vee$") da forma usual. Seja $\phi, \psi \in ForCTL(\mathcal{P})$:

- $\exists \mathsf{F}\phi = \exists[\mathtt{true}\mathsf{U}\phi]$
- $\forall \mathsf{X}\phi = \neg\exists \mathsf{X}\neg\phi$
- $\forall \mathsf{G}\phi = \neg\exists \mathsf{F}\neg\phi$
- $\forall[\phi \mathsf{U} \psi] = \neg\exists[\neg\psi \mathsf{U}(\neg\phi \wedge \neg\psi)] \wedge \neg\exists \mathsf{G}\neg\psi$
- $\forall \mathsf{F}\phi = \neg\exists \mathsf{G}\neg\phi$.

O significado de fórmulas CTL é dado em termos de modelos Kripke e caminhos, que também chamamos de execuções. Assim sendo, necessitaremos definir "caminho em uma estrutura Kripke" para dar semântica as fórmulas da linguagem.

Definição 3.1.6 *(Caminho ou Execução em uma estrutura Kripke)*

Seja $\mathfrak{M} = \langle S, S_0, R\rangle$ uma estrutura Kripke. Um caminho π em $\mathfrak{M}$ se caracteriza por uma seqüencia infinita de estados, $\pi = s_0, s_1, s_2, \cdots$, tal que para todo $i \geq 0$, $(s_i, s_{i+1}) \in R$.

Usamos $\pi(i)$ para denotar o estado de ordem i de π.

Definição 3.1.7 *(Relação de Satisfação para fórmulas CTL)*

Seja $\mathfrak{M} = \langle S, S_0, R, \mathcal{P}, \mathcal{L}\rangle$ um modelo Kripke. A relação de satisfação para fórmulas CTL é definida da seguinte forma:

$\mathfrak{M}, s \models p$	*se e somente se*	$p \in \mathcal{L}(s)$
$\mathfrak{M}, s \models \neg\phi$	*se e somente se*	$\mathfrak{M}, s \not\models \phi$
$\mathfrak{M}, s \models \phi \vee \psi$	*se e somente se*	$\mathfrak{M}, s \models \phi$ *ou* $\mathfrak{M}, s \models \psi$
$\mathfrak{M}, s \models \exists \mathsf{X}\phi$	*se e somente se*	*existe um caminho* π *iniciado em* s *tal que* $\mathfrak{M}, \pi(1) \models \phi$
$\mathfrak{M}, s \models \exists \mathsf{F}\phi$	*se e somente se*	*existe um caminho* π *iniciado em* s *e existe um* $k \geq 0$ *tal que* $\mathfrak{M}, \pi(k) \models \phi$
$\mathfrak{M}, s \models \exists \mathsf{G}\phi$	*se e somente se*	*existe um caminho* π *iniciado em* s *e para todo* $k \geq 0$, $\mathfrak{M}, \pi(k) \models \phi$
$\mathfrak{M}, s \models \exists[\phi \mathsf{U} \psi]$	*se e somente se*	*existe um caminho* π *iniciado em* s *tal que existe um* $k \geq 0$ *para o qual* $\mathfrak{M}, \pi(k) \models \psi$ *e para todo* j, $0 \leq j \leq k$, $\mathfrak{M}, \pi(j) \models \phi$

3.1.3 Verificação de Modelos

Dado um modelo Kripke $\mathfrak{M} = \langle S, S_0, R, \mathcal{P}, \mathcal{L} \rangle$ que represente um sistema concorrente com suas propriedades de interesse e uma fórmula de CTL f expressando uma especificação desejada, o *problema de model checking* (ou de checagem de modelos) é encontrar um conjunto de estados em S que satisfaça f: $\{s \in S | \mathfrak{M}, s \models f\}$ [PCG00]. Em outras palavras, o modelo de estados e transições que está por baixo do modelo Kripke é checado para se descobrir se este é um modelo para a especificação escrita em CTL.

Normalmente alguns estados são designados estados iniciais, e dizemos que o sistema satisfaz a especificação uma vez que todos os estados iniciais estão no conjunto encontrado. Formalmente, $\mathfrak{M}, S_0 \models f$ significa $\forall_{s_0 \in S_0} \mathfrak{M}, s_0 \models f$.

Uma fórmula CTL f pode ser identificada com um conjunto de estados em um dado modelo $\mathfrak{M}$, especificamente com os estados $Q_{\mathfrak{M}} \subseteq S$ que satisfazem a fórmula: $Q_{\mathfrak{M}}(f) = \{s | \mathfrak{M}, s \models f\}$. A checagem de modelos de uma fórmula CTL implica então um processo de manipulação de estados: $S_0 \subseteq Q_{\mathfrak{M}}(f)$.

Seja $\mathfrak{M} = \langle S, S_0, R, \mathcal{P}, \mathcal{L} \rangle$ um modelo Kripke, e f uma fórmula CTL. O algoritmo para determinar quais estados de S satisfazem f opera rotulando cada estado de S com o conjunto *label*(s) de subfórmulas de f que são verdadeiras em s. Inicialmente, *label*(s) é apenas $\mathcal{L}(s)$. O algoritmo passa então por uma série de estágios. Durante o estágio de ordem i, subfórmulas com $i - 1$ operadores de CTL são processadas. Quando uma subfórmula é processada, ela é adicionada ao conjunto de rótulos de cada estado em que

ela é verdadeira. Quando o algoritmo termina, teremos que $\mathfrak{M}, s \models f$ se e somente se $f \in \mathit{label}(s)$.

Relembrando que uma fórmula CTL pode envolver somente os operadores $\neg$, $\vee$, $\exists\mathsf{X}$, $\exists\mathsf{G}$ e $\exists\mathsf{U}$, apenas o tratamento destes seis casos é o suficiente para estágios intermediários do algoritmo, dependendo se f é uma fórmula atômica ou de uma das seguinte formas: $\neg f_1$, $f_1 \vee f2$, $\exists\mathsf{X} f_1$, $\exists[f_1 \mathsf{U} f_2]$, $\exists\mathsf{G} f_1$.

O processo de rotulação prossegue da seguinte forma para cada um dos casos :

- As fórmulas atômicas já estão tratadas, uma vez que os estados já estão rotulados com fórmulas atômicas no modelo Kripke;
- $\neg f_1$, rotulamos com $\neg f_1$ os estados que não estão rotulados com a fórmula f_1.
- $f_1 \vee f_2$, rotulamos com $f_1 \vee f_2$ os estados que estão rotulados com f_1, com f_2 ou com ambas;
- $\exists\mathsf{X} f_1$, rotulamos os estados que tem um sucessor na relação de transição rotulado com f_1;
- $\exists[f_1 \mathsf{U} f_2]$, primeiro é necessário encontrar todos os estados rotulados com f_2, e depois trabalhamos caminhando para trás na relação de transição, utilizando a inversa de R e procurando todos os estados $t \in S$ que podem ser alcançados por um caminho onde cada estado do caminho esteja rotulado com f_1. Uma vez encontrados, rotular os estados t com $\exists[f_1 \mathsf{U} f_2]$;

O tratamento do caso $\exists\mathsf{G} f_1$ baseia-se na decomposição do grafo em componentes fortemente conexas não triviais, conforme veremos a seguir.

Definição 3.1.8 *(Componente Fortemente Conexa e Componente Fortemente Conexa não trivial)*

Uma componente fortemente conexa C de um grafo é um subgrafo maximal tal que todo nó em C é alcançável de qualquer outro nó em C através de um caminho direcionado totalmente contido em C. C é não trivial se e somente se possui mais de um nó ou se possui um nó com um laço para si mesmo (self-loop).

- $\exists \mathsf{G} f_1$, primeiro criamos um modelo restrito baseado em $\mathfrak{M}$, $\mathfrak{M}'$, onde $\mathfrak{M}' = \langle S', S'_0, R', \{\sim_k\}'_{1 \leq k \leq j}, \cup_{k=1}^{j} \mathcal{P}_k, \mathcal{L}' \rangle$ é obtido a partir de $\mathfrak{M}$ removendo de S todos os estados onde f_1 não é verdadeiro e restringindo R e L em acordo. R' pode não ser total. Então, particiona-se o grafo (S', R') em componentes fortemente conexas. Depois, encontra-se os estados que pertencem as componentes não triviais, e trabalha-se com eles seguindo a relação de transição de trás para a frente, usando a inversa de R', para encontrar todos os estados alcançáveis por um caminho no qual cada estado está rotulado com f_1. Finalmente, rotular estes estados com $\exists \mathsf{G} f_1$;

O algoritmo 2 mostra o processo de rotulação de estados com o operador $\exists \mathsf{U}$. Este processo requer tempo $O(|S| + |R|)$. O mesmo tempo é requerido para o processo de rotulação correspondente ao operador $\exists \mathsf{G} f$, mostrado no algoritmo 3.

Algorithm 2: CheckEU(f_1, f_2)

```
begin
    T := {s | f2 ∈ label(s)}
    forall s ∈ T do
        label(s) := label(s) ∪ {∃[f1 U f2]}
    end
    while T ≠ ∅ do
        choose s ∈ T
        T := T \ {s}
        forall t such that R(t, s) do
            if ∃[f1 U f2] ∉ label(t) and f1 ∈ label(t) then
                label(t) :=label(t) ∪ {∃[f1 U f2]}
                T := T ∪ {t}
            end
        end
    end
end
```

Algorithm 3: CheckEG(f_1)

begin
  $S' := \{s | f_1 \in \text{label}(s)\}$
  $SCC := \{C | C$ é uma componente fortemente conexa não trivial de $S'\}$
  $T := \bigcup_{C \in SCC} \{s | s \in C\}$
  forall $s \in T$ **do**
    label(s) := label$(s) \cup \{\exists \mathsf{G} f_1\}$
  end
  while $T \neq \emptyset$ **do**
    choose $s \in T$
    $T := T \setminus \{s\}$
    forall t *such that* $t \in S'$ *and* $R(t, s)$ **do**
      if $\exists \mathsf{G} f_1 \notin$ *label*(t) **then**
        label(t) :=label$(t) \cup \{\exists \mathsf{G} f_1\}$
        $T := T \cup \{t\}$
      end
    end
  end
end

Para o tratamento de uma fórmula arbitrária de f de CTL, aplica-se sucessivamente o processo de rotulação de estados previamente descrito para as subfórmulas de f, começando pelas menores e mais aninhadas, e vamos sucessivamente processando subfórmulas a cada passo maiores, até cobrir a fórmula f completa. Procedendo desta maneira, fica garantido que quando chega a hora de processar uma certa subfórmula g de f todas as subfórmulas de g já tenham sido processadas. Como cada processo de rotulação de uma subfórmula requer no pior caso $O(|S| + |R|)$, e como f tem no máximo $|f|$ subfórmulas distintas, o algoritmo completo requer tempo $O(f \cdot |S| + |R|)$.

Teorema 3.1.1 *Existe um algoritmo para determinar se uma fórmula f de CTL é verdadeira em um estado s de uma estrutura $\mathfrak{M} = \langle S, S_0, R, \mathcal{P}, \mathcal{L} \rangle$ que executa em tempo $O(f \cdot |S| + |R|)$ [PCG00].*

3.2 Modelagem e verificação de sistemas probabilísticos de tempo discreto

Existe um grande número de modelos probabilísticos, dos quais consideramos para o escopo desta tese apenas alguns que são variantes do modelo tradicional de Cadeia de Markov com espaço discreto de estados. A classe de modelos mais simples são as Cadeias de Markov de Tempo Discreto, que apresenta escolhas probabilísticas e não apresenta não determinismo. A segunda classe de modelos que iremos considerar são os Processos de Decisão de Markov, que possibilita tanto escolhas probabilísticas quando não determinismo. Os Processos de Decisão de Markov são modelos que surgem naturalmente da composição paralela de modelos do tipo Cadeias de Markov de Tempo Discreto, e são particularmente apropriados no contexto de sistemas distribuídos [RKNP04]. Por esta razão, focaremos nesta segunda classe de modelos. Ambas as classes são modelos de tempo discreto. Um modelo para tempo contínuo será apresentado na seção 3.3.

Após a introdução dos modelos, apresentaremos uma linguagem para expressar propriedades relevantes e em seguida os algoritmos de verificação correspondentes.

3.2.1 Modelagem: Cadeia de Markov de Tempo Discreto e Processo de Decisão de Markov

Uma Cadeia de Markov de Tempo Discreto (*Discrete-Time Markov Chain* – DTMC) é uma família de variáveis randômicas $\{X(k)|k = 0, 1, 2, \cdots\}$, onde as observações são feitas em tempos discretos. Os valores assumidos por $X(k)$ são os estados do sistema. O conjunto de todos os possíveis estados, conhecido como espaço de estados, é discreto e restringiremos o modelo para considerar apenas espaços de estado finitos. Uma DTMC tem a propriedade de Markov, também conhecida como propriedade da ausência de memória, que significa que o estado do sistema no instante k depende apenas do estado do sistema no instante anterior – isto é, para todo inteiro $k \geq 0$ e estados $s_0, \cdots, s_k$:

$$P[X(k) = s_k | X(k-1) = s_{k-1}, \cdots, X(0) = s_0] = P[X(k) = s_k | X(k-1)]$$

Além disso, consideraremos apenas DTMCs homogêneas, para as quais a probabilidade de ocorrência de uma transição independe do tempo em que a transição ocorre. Isso

significa que para descrever uma DTMC é suficiente dizer a probabilidade de que uma transição ocorra entre dois estados.

Definição 3.2.1 *(Cadeia de Markov de Tempo Discreto)*
Uma DTMC é uma tupla $\mathcal{D} = (S, s_0, \mathbf{P})$ *onde:*

- S *é um conjunto finito de estados;*
- $s_0 \in S$ *é o estado inicial;*
- $\mathbf{P} : S \times S \to [0, 1]$ *é uma matriz de transição de probabilidades, tal que:* $\sum_{s' \in S} \mathbf{P}(s, s') = 1$ *para todos os estados* $s \in S$.

Cada elemento $\mathbf{P}(s, s')$ da matriz de transição de probabilidades corresponde à probabilidade de realizar uma transição do estado s para o estado s', isto é, para qualquer $k \geq 0$:

$$\mathbf{P}(s, s') = P[X(k+1) = s' | X(k) = s].$$

A soma das probabilidades associadas a transições emanando de um mesmo estado deve ser necessariamente 1. Estados terminais, isto é, aqueles de onde o sistema não pode mudar para outro estado, são modelados com a adição de uma transição para si mesmo com probabilidade 1 (auto-loop).

Para que se possa utilizar DTMCs para fazer análise de propriedades, precisamos rotular o modelo com informações adicionais. A função de rotulação $L : S \to 2^{AP}$ mapeia estados em conjuntos de proposições oriundas de um conjunto AP. Utilizamos este conjunto de proposições atômicas para para rotular a DTMC com propriedades de interesse. Também rotulamos a DTMC com o custo , uma função $\mathbf{C} : S \times S \to \mathbb{R}^{\geq 0}$ que mapeia cada par de estados em valores não negativos. Intuitivamente, $\mathbf{C}(s, s')$ denota o custo de realizar uma transição de s para s'.

As DTMCs são o modelo probabilístico básico, porém não o mais adequado para modelar sistemas multi-agentes. Por isso, não detalharemos mais o processo de análise e verificação de propriedades correspondente a essa classe de modelos. Consideraremos a seguir um modelo que pode ser considerado uma generalização das DTMCs, os processos de decisão de Markov.

Um processo de decisão de Markov (*Markov Decision Process* – MDP) é um modelo que descreve comportamentos probabilísticos e puramente não determinísticos (sem probabilidade associada), em um ambiente onde a passagem do tempo é medida em passos. Este modelo se adeqüa muito bem ao modelo de sistemas concorrentes, uma vez que o não determinismo é reconhecidamente um recurso valioso para modelar concorrência. Através dos MDPs podemos descrever comportamentos de vários sistemas probabilísticos operando em conjunto.

Definição 3.2.2 *(Ditribuições Probabilísticas sobre um conjunto)*

Uma Distribuição Probabilística sobre um conjunto S é o conjunto das funções $\mu : S \to [0,1]$ tais que $\sum_{s \in S} \mu(s) = 1$.

Definição 3.2.3 *(Processo de Decisão de Markov)*

Um Processo de Decisão de Markov é uma n-upla $\mathcal{M} = (S, s_0, Act, R)$, onde:

- *S é um conjunto finito de estados;*
- *$s_0 \in S$ é o estado inicial;*
- *Act é um conjunto de ações;*
- *$R : S \to 2^{Act \times Dist(S)}$ é uma função de transição probabilística, onde $Dist(S)$ é o conjunto de todas as ditribuições probabilísticas sobre o conjunto S.*

É interessante notar que em um MDP existe uma escolha não determinística sobre as distribuições probabilísticas rotuladas pelas ações em Act, denotadas por $|R(s)|$. Uma vez que um par $(a, \mu) \in R(s)$ seja escolhido, a ação a é executada e a probabilidade do próximo estado ser s' é $\mu(s')$. Ainda assim, o modelo satisfaz a propriedade de Markov, que diz que o próximo estado do sistema depende apenas do estado atual.

Para o propósito de análise de sistemas, enriquecemos os MDPs com proposições:

Definição 3.2.4 *(Processo de Decisão de Markov rotulado)*

Seja AP um conjunto de proposições atômicas. Um MDP rotulado é um trio ordenado $(\mathcal{M}, \mathcal{L}, c)$ onde:

- *$\mathcal{M}$ é um MDP;*

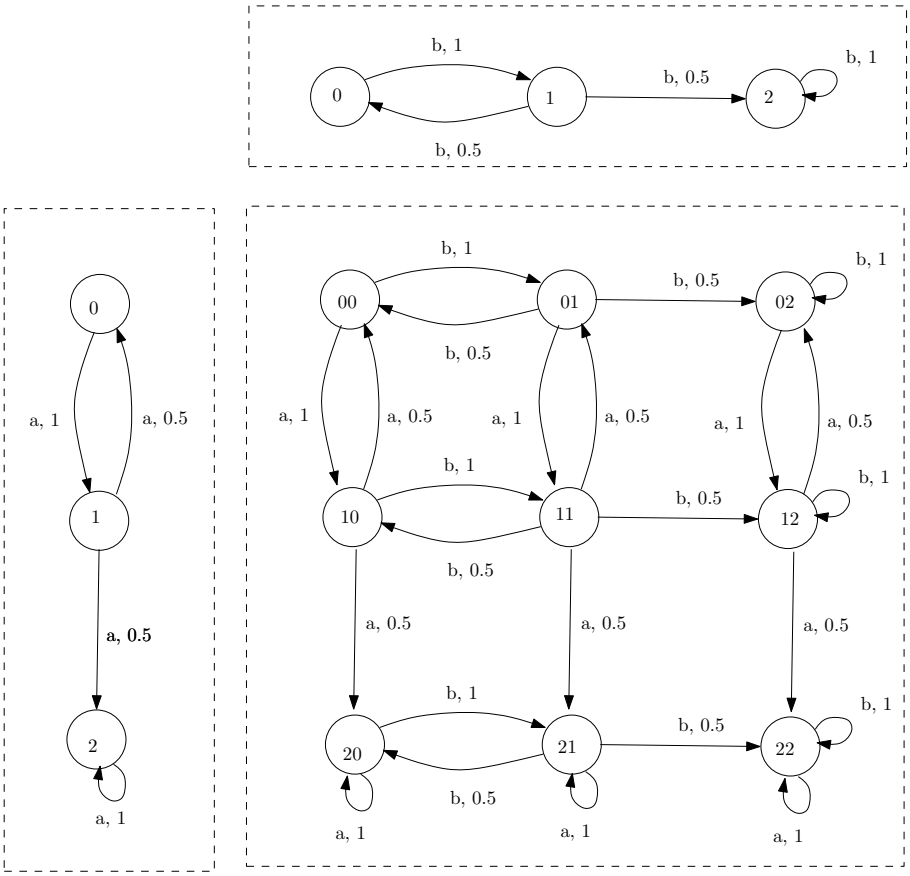


Figura 3.1: Um MDP formado pela composição de duas DTMCs.

- $\mathcal{L} : 2^{AP}$ *é a função que associa a cada estado um subconjunto de proposições;*
- $\mathbf{c} : S \times Act \rightarrow \mathbb{R}^{\geq 0}$ *é a função de custo, que associa custos não negativos às transições.*

A figura 3.1 mostra exemplos de DTMCs e um MDP, e de como esta classe de modelos MDP pode ser utilizada para modelar composição paralela de sistemas probabilísticos da classe DTMC.

Definição 3.2.5 *(Caminho em um MDP)*

Um caminho w *em um MDP é uma seqüencia da forma* $s_0 \xrightarrow{(a_1,\mu_1)} s_1 \xrightarrow{(a_2,\mu_2)} s_2 \cdots$, *onde* $s_i \in S$, $(a_{i+1}, \mu_{i+1}) \in R(s_i)$, *e* $\mu_{i+1} > 0$ *para todo* $i \geq 0$.

Usaremos $w(i)$ para denotar o estado de ordem i no caminho w, $|w|$ o tamanho do caminho w, e $last(w_{fin})$ o último estado do caminho finito w_{fin}. $R(w, i)$ denotará a ação

de ordem i do caminho w. E $Path_s^{fin}$, $Path_s$ denotam o conjunto de todos os caminhos finitos e infinitos iniciados em s, respectivamente.

Para definir um caminho em um MDP, ambas as escolhas probabilísticas e não determinísticas tem que ser resolvidas. Por conta disso, assumimos que as escolhas não determinísticas são feitas por um *adversário*, que pode ser visto como um gerenciador de tarefas ou uma política de escolha.

Definição 3.2.6 *(Adversário em um MDP)*

Um Adversário A de um MDP $\mathcal{M}$ é uma função que mapeia todo caminho w_{fin} de $\mathcal{M}$ em um elemento $A(w_{fin})$ do conjunto $R(last(w_{fin}))$.

$Adv_{\mathcal{M}}$ denotará o conjunto de todos os possíveis adversários de $\mathcal{M}$, e para um adversário A, $Path_s^A$ denota o subconjunto de $Path_s$ que corresponde a A.

O comportamento de um MDP sob um determinado adverário A é puramente probabilístico. Para falar sobre o comportamento de um MDP sob um determinado adversário, é necessário determinar a probabilidade de que certos caminhos sejam tomados. Para isso, definimos, para cada estado $s \in S$, uma medida probabilística $Prob_s^A$ sobre $Path_s^A$.

Começamos por definir uma medida probabilística para caminhos finitos.

Definição 3.2.7 *(Medida probabilística de um caminho finito iniciado no estado s, dado um adversário A)*

Para cada caminho finito $w_{fin} \in Path_s^{A,fin}$, $P_s^A(w_{fin})$ é definido como:

$$P_s^A(w_{fin}) \stackrel{def}{=} \begin{cases} 1 & \text{if } n = 0 \\ P(w(0), w(1)) \cdots P(w(n-1), w(n)) & \text{caso contrário} \end{cases}$$

onde $n = |w_{fin}|$ e para dois caminhos finitos $w, w' \in Path_s^{A,fin}$:

$$P^A(w, w') = \begin{cases} \mu(s') & \text{se } w' \text{ é da forma } w \xrightarrow{A(w)} s' \text{ e } A(w) = (a, \mu) \\ 0 & \text{caso contrário} \end{cases}$$

Através da definição de conjuntos cilíndricos, iremos estender a definição de medida probabilística para caminhos infinitos.

Definição 3.2.8 *(Conjunto Cilíndrico de um caminho finito)*

Um conjunto cilíndrico $C(w_{fin})$ *é definido por:*

$$C(w_{fin}) \stackrel{def}{=} \{w \in Path_s^A | w_{fin} \textit{é prefixo de } w\}$$

Ou seja, $C(w_{fin})$ *é o conjunto de todos os caminhos infinitos com prefixo* w_{fin}*.*

Seja Σ_s^A a menor σ-algebra em $Path_s^A$ que contém todos os conjuntos $C(w_{fin})$, onde w_{fin} abrange caminhos em $Path_s^{A,fin}$. Usaremos Σ_s para definir a medida probabilística $Prob_s$ sobre $Path_s^A$.

Definição 3.2.9 *(Medida Probabilística* $Prob_s^A$ *sobre* $Path_s^A$*, dado* Σ_s^A*)*

A medida probabilística $Prob_s^A$ *dado* Σ_s^A *é definida como a única medida tal que:* $Prob_s^A(C(w_{fin})) = P_s^A(w_{fin})$*, para todo* $w_{fin} \in Path_s^{A,fin}$*.*

Assim podemos quantificar a probabilidade que um MDP sobre o controle de um adversário A se comporte de uma determinada maneira identificando o conjunto dos caminhos que satisfazem a especificação, e assumindo que este seja um conjunto mensurável, usando a medida de probabilidade associada $Prob_s^A$.

Definimos também para um adverário A, o custo esperado de se alcançar um conjunto de estados alvo F, fazendo uso de $Prob_s^A$:

Definição 3.2.10 *(Custo Esperado do conjunto de estados alvo* F*)*

O custo esperado de se alcançar um conjunto de estados alvo F *é definido através de* $E_s^A(cost(F))$*, onde* E_s^A *é a esperança usual com respeito a medida* $Prob_s^A$ *e para qualquer* $w \in Path_s^A$*:*

$$cost(F)(w) \stackrel{def}{=} \begin{cases} \sum_{i=0}^{min\{i|w(i)\in F\}} \mathbf{c}(R(w, i-1)) & \textit{se } \exists i \in \mathbb{N} \cdot w(i) \in F \\ \infty & \textit{caso contrário} \end{cases}$$

3.2.2 Especificação: Lógica probabilística de computação ramificada PCTL

Para escrever especificações de MDPs, usa-se uma extensão probabilística da lógica CTL, que chamaremos de lógica probabilística de tempo ramificado, *Probabilistic Computation Tree Logic* – PCTL [HJ94], [ASB95], [RKNP04].

Definição 3.2.11 *(Sintaxe de fórmulas PCTL)*

A sintaxe de fórmulas de PCTL é:

$\phi ::= true \mid a \mid \neg\phi \mid \phi \wedge \phi \mid \mathcal{P}_{\bowtie p}[\psi] \mid \mathcal{E}_{\bowtie c}[\phi]$, $\psi ::= \phi \mid \phi \mathcal{U}^{\leq k} \phi \mid \phi \mathcal{U} \phi$

onde a é uma proposição atômica, $\bowtie \in \{\leq, <, \geq, >\}$, $p \in [0,1]$, $c \in \mathbb{R}^{\geq 0}$ e $k \in \mathbb{N}$

A sintaxe é descrita em termos de fórmulas de estado ϕ e de fórmulas de caminho ψ, que são avaliadas respectivamente sobre estados e caminhos, porém propriedades de um MDP serão sempre expressas através de fórmulas de estado.

Definição 3.2.12 *(Relação de Satisfação para fórmulas de PCTL)*

Seja $\mathcal{M} = (S, S_0, Act, R, \mathcal{L}, c)$ um MDP rotulado. A relação de satisfação para fórmulas PCTL é definida da seguinte forma:

$s \models true$	*para todo*	$s \in S$
$s \models a$	*se e somente se*	$a \in \mathcal{L}(s)$
$s \models \neg\phi$	*se e somente se*	$s \not\models \phi$
$s \models \phi_1 \wedge \phi_2$	*se e somente se*	$s \models \phi_1$ *e* $s \models \phi_2$
$s \models \mathcal{P}_{\bowtie p}[\psi]$	*se e somente se*	$p_s^A(\psi) \bowtie p$, *para todo* $A \in Adv_{\mathcal{M}}$
$s \models \mathcal{E}_{\bowtie c}[\phi]$	*se e somente se*	$e_s^A(\phi) \bowtie c$, *para todo* $A \in Adv_{\mathcal{M}}$

onde para todo adversário $A \in Adv_{\mathcal{M}}$:

$p_s^A(\psi) \stackrel{def}{=} Prob_s^A(\{w \in Path_s^A | w \models \psi\})$, $e_s^A(\phi) \stackrel{def}{=} E_s^A(cost(\{s' \in S | s' \models \phi\})$

e para cada caminho $w \in Path$:

$w \models \phi$	*se e somente se*	$w(1) \models \phi$,
$w \models \phi_1 \mathcal{U}^{\leq k} \phi_2$	*se e somente se*	$\exists i \leq k \cdot (w(i) \models \phi_2 \wedge w(j) \models \phi_1, \forall j < i)$,
$w \models \phi_1 \mathcal{U} \phi_2$	*se e somente se*	$\exists k \geq 0 \cdot w \models \phi_1 \mathcal{U}^{\leq k} \phi_2$.

Vale ressaltar que para qualquer fórmula de caminho ψ, estado $s \in S$ e adversário A, o conjunto de caminhos $\{w \in Path_s^A | w \models \psi\}$ é mensurável [Var85].

Os operadores $false$, $\vee$ e $\rightarrow$ podem ser definidos em função de $true$, $\neg$ e $\wedge$.

Podemos também definir operadores de fórmulas de caminho $\diamond$ e $\diamond^{\leq k}$:

$$\mathcal{P}_{\bowtie p}[\diamond\phi] \equiv \mathcal{P}_{\bowtie p}[true \mathcal{U} \phi], \qquad \mathcal{P}_{\bowtie p}[\diamond^{\leq k}\phi] \equiv \mathcal{P}_{\bowtie p}[true \mathcal{U}^{\leq k} \phi]$$

E os operadores $\Box$ e $\Box^{\leq k}$, usando o fato de que para um estado s, adversário A e fórmula de caminho ψ, $p_s^A(\neg\psi) = 1 - p_s^A(\psi)$:

$$\mathcal{P}_{\bowtie p}[\Box\phi] \equiv \mathcal{P}_{\overline{\bowtie} 1-p}[\Diamond\neg\phi], \qquad \mathcal{P}_{\bowtie p}[\Box^{\leq k}\phi] \equiv \mathcal{P}_{\overline{\bowtie} 1-p}[\Diamond^{\leq k}\neg\phi],$$

onde $\overline{\leq} \equiv \geq, \overline{<} \equiv >, \overline{\geq} \equiv \leq$ e $\overline{>} \equiv <$.

Ao escrever uma especificação pode ser útil considerar a *existência* de um adversário, ao invés de quantificar sobre todos os adversários. Isso pode ser feito por meio de tradução para a propriedade dual, por exemplo:

- $p_s^A(\psi) > p$ para algum adversário $A \Leftrightarrow \neg\mathcal{P}_{\leq p}[\psi]$,
- $p_s^A(\psi) \leq p$ para algum adversário $A \Leftrightarrow \neg\mathcal{P}_{>p}[\psi]$,
- $e_s^A(\psi) < p$ para algum adversário $A \Leftrightarrow \neg\mathcal{E}_{\geq p}[\psi]$,
- $e_s^A(\psi) \geq p$ para algum adversário $A \Leftrightarrow \neg\mathcal{E}_{<p}[\psi]$.

Fazendo uso desta idéia, podemos construir um operador análogo de $\exists$F em CTL: $\exists \Diamond \phi \equiv \neg\mathcal{P}_{\geq 0}[\Diamond\phi]$, que avalia a existência de um adversário para o qual a probabilidade de atingir um estado satisfazendo ϕ é maior que zero.

3.2.3 Verificação de especificações em PCTL sobre modelos MDP

O algoritmo de verificação de modelos para PCTL sobre MDPs recebe como entrada um MDP rotulado $(S, s_0, Act, R, \mathcal{L}, c)$ e uma fórmula PCTL ϕ, e retorna o conjunto $Sat(\phi) \subseteq S$ de estados que satisfazem ϕ [BdA95, CY88, CY90, CY95, RKNP04].

O algoritmo trabalha checando se cada estado do modelo satisfaz ϕ. A estrutura geral do algoritmo é bem similar ao algoritmo de checagem de modelos para CTL: a fórmula ϕ é trabalhada de "dentro para fora", ou seja, quebrada em subfórmulas até que se chegue a suas componentes atômicas. Primeiro são trabalhadas as componentes atômicas (subfórmulas de tamanho 0), depois as fórmulas envolvendo operadores e subfórmulas de tamanho 0 (subfórmulas de tamanho 1), e assim por diante, num processo recursivo, de forma que no passo n são trabalhadas subfórmulas de tamanho n, com o uso dos resultados obtidos de todas as subfórmulas de tamanho menor ou igual a $n - 1$ que já foram

trabalhadas em passos anteriores. Ao final, a última subfórmula a ser trabalhada será a própria fórmula ϕ.

O algoritmo pode ser resumido da seguinte forma:

- $Sat(true) = S$,
- $Sat(false) = \emptyset$,
- $Sat(a) = \{s | a \in \mathcal{L}(S)\}$,
- $Sat(\neg\phi) = S \setminus Sat(\phi)$,
- $Sat(\phi_1 \wedge \phi_2 = Sat(\phi_1) \cap Sat(\phi_2)$,
- $Sat(\phi_1 \vee \phi_2 = Sat(\phi_1) \cup Sat(\phi_2)$,
- $Sat(\phi_1 \implies \phi_2) = (S \setminus Sat(\phi_1)) \cup Sat(\phi_2)$,
- $Sat(\mathcal{P}_{\bowtie p}[\psi]) = \{s \in S | p_s^{max}(\psi) \bowtie p\}$, $\quad Sat(\mathcal{E}_{\bowtie c}[\phi]) = \{s \in S | e_s^{max}(\phi) \bowtie c\}$, se $\bowtie$ é $\leq$ ou $<$,
- $Sat(\mathcal{P}_{\bowtie p}[\psi]) = \{s \in S | p_s^{min}(\psi) \bowtie p\}$, $\quad Sat(\mathcal{E}_{\bowtie c}[\phi]) = \{s \in S | e_s^{min}(\phi) \bowtie c\}$, se $\bowtie$ é $\geq$ ou $>$,

onde:

$$p_s^{max}(\psi) \stackrel{\text{def}}{=} \sup_{A \in Adv_{\mathcal{M}}} [p_s^A(\psi)], \qquad e_s^{max}(\phi) \stackrel{\text{def}}{=} \sup_{A \in Adv_{\mathcal{M}}} [e_s^A(\phi)]$$

$$p_s^{min}(\psi) \stackrel{\text{def}}{=} \inf_{A \in Adv_{\mathcal{M}}} [p_s^A(\psi)], \qquad e_s^{min}(\phi) \stackrel{\text{def}}{=} \inf_{A \in Adv_{\mathcal{M}}} [e_s^A(\phi)]$$

O algoritmo para os operadores não probabilísticos de PCTL é semelhante aos casos equivalentes de CTL. Para os operadores $\mathcal{P}_{\bowtie p}(\psi)$ e $\mathcal{E}_{\bowtie c}(\phi)$ é necessário determinar se $p_s(\psi)$ (ou $e_s(\phi)$) satisfaz o limite $\bowtie p$ (ou $\bowtie c$) para todos os adversários A. Fazemos isso computando a probabilidade mínima ou máxima (ou o custo esperado) sobre todos os adversários, dependendo da relação especificada pelo operador $\bowtie$.

Seguem os respectivos algoritmos para fazer esta computação para os casos: $\mathcal{P}_{\bowtie p}[\phi]$, $\mathcal{P}_{\bowtie p}[\phi_1 \mathcal{U}^{leqk} \phi_2]$, $\mathcal{P}_{\bowtie p}[\phi_1 \mathcal{U} \phi_2]$, $\mathcal{E}_{\bowtie c}[\phi]$.

Algoritmo para o operador $\mathcal{P}_{\bowtie p}[\phi]$

Para verificar a satisfação de uma fórmula deste tipo é necessário computar as probabilidades $p_s^{max}(\phi)$ ou $p_s^{min}(\phi)$. Como o algoritmo trabalha recursivamente, ao chegar a hora de processar a subfórmula $p_s^{min}(\phi)$, a subfórmula ϕ já terá sido processada, e os estados em que ela é satisfeita já serão conhecidos. Para cada estado s precisamos então considerar apenas as transições imediatas para estados seguintes a s:

$$p_s^{max}(\phi) = \max_{(a,\mu)\in R(s)} \Big\{ \sum_{s'\in Sat(\phi)} \mu(s') \Big\}, \qquad p_s^{min}(\phi) = \min_{(a,\mu)\in R(s)} \Big\{ \sum_{s'\in Sat(\phi)} \mu(s') \Big\}$$

Seja m o número total de escolhas não determinísticas em todos os estados do MDP $m = \sum_{s\in S} |R(s)|$. Podemos representar a função R como uma matriz RM de dimensões $m \times |S|$, onde cada linha da matriz corresponde a uma única escolha não determinística. Com isso, cada linha da matriz corresponde a uma distribuição para as escolhas não determinísticas de $R(s)$.

Podemos assumir, sem perda de generalidade, a existência de um vetor indexado por estados, $\underline{\phi}$, tal que $\underline{\phi}(s)$ é igual a 1 se $s \models \phi$ e igual a zero caso contrário.

O funcionamento do algoritmo transcorre em dois passos da seguinte forma:

1. Primeiramente, uma multiplicação da matriz construída RM pelo vetor $\underline{\phi}$, que computa a soma para cada escolha não determinística, produzindo um vetor de tamanho m;

2. Depois, uma operação para selecionar, deste vetor, o valor máximo ou mínimo para cada estado, reduzindo o tamanho de m para S.

O vetor final tem tamanho $|S|$ e guarda para todo estado s, na posição indexada por s, o valor de p_s^{max} ou p_s^{min}, dependendo da operação que foi feita (max ou min).

Algoritmo para o operador $\mathcal{P}_{\bowtie p}[\phi_1 \mathcal{U}^{\leq k} \phi_2]$

É necessário calcular $p_s^{max}(\phi_1 \mathcal{U}^{\leq k} \phi_2)$ ou $p_s^{min}(\phi_1 \mathcal{U}^{\leq k} \phi_2)$.

Para isso, dividimos o conjunto S em três subconjuntos:

$$S^{no} = S \setminus (Sat(\phi_1) \cup Sat(\phi_2)), \qquad S^{yes} = Sat(\phi_2), \qquad S^{?} = S \setminus (S^{no} \cup S^{yes})$$

onde, para $s \in S^{no}$ e $s \in S^{yes}$, $p_s^{max}(\phi_1 \mathcal{U}^{\leq k} \phi_2) = p_s^{min}(\phi_1 \mathcal{U}^{\leq k} \phi_2)$ são trivialmente iguais a 0 ou 1, respectivamente.

Probabilidades para os demais estados em $S^{?}$ são definidas recursivamente:

- Se $k = 0$, então $p_s^{max}(\phi_1 \mathcal{U}^{\leq k} \phi_2) = p_s^{min}(\phi_1 \mathcal{U}^{\leq k} \phi_2) = 0$
- Se $k > 0$:

$$p_s^{max}(\phi_1 \mathcal{U}^{\leq k} \phi_2) = \underset{(a,\mu) \in R(s)}{max} \Big\{ \sum_{s' \in S} \mu(s') \cdot p_{s'}^{max}(\phi_1 \mathcal{U}^{\leq k-1} \phi_2) \Big\},$$

$$p_s^{min}(\phi_1 \mathcal{U}^{\leq k} \phi_2) = \underset{(a,\mu) \in R(s)}{min} \Big\{ \sum_{s' \in S} \mu(s') \cdot p_{s'}^{min}(\phi_1 \mathcal{U}^{\leq k-1} \phi_2) \Big\}.$$

Como no caso anterior, podemos assumir que R é representada por uma matriz RM, e dessa forma a computação de $p_s^{max}(\phi_1 \mathcal{U}^{\leq k} \phi_2)$ ou $p_s^{min}(\phi_1 \mathcal{U}^{\leq k} \phi_2)$ pode ser feita em k interações, cada uma similar ao processo descrito anteriormente – cada interação envolverá uma multiplicação matriz-vetor e uma operação máximo ou mínimo.

Algoritmo para o operador $\mathcal{P}_{\bowtie p}[\phi_1 \mathcal{U} \phi_2]$

Para este operador é necessário computar uma entre as probabilidades $p_s^{max}(\phi_1 \mathcal{U} \phi_2)$ ou $p_s^{min}(\phi_1 \mathcal{U} \phi_2)$, dependendo de $\bowtie$.

O primeiro passo é montar os conjuntos S^{no} e S^{yes} que contém todos os estados com probabilidades igual a 0 ou 1, respectivamente. Isso será feito usando um algoritmo baseado em grafos, que depende se estamos calculando as probabilidades máxima ou mínima.

Para a computação de $p_s^{max}(\phi_1 \mathcal{U} \phi_2)$:

$$S^{no} = PROB0A(Sat(\phi_1), Sat(\phi_2)),$$

$$S^{yes} = PROB1E(Sat(\phi_1), Sat(\phi_2)),$$

$$S^{?} = S \setminus (S^{no} \cup S^{yes})$$

e para $p_s^{min}(\phi_1 \mathcal{U} \phi_2)$:

$$S^{no} = PROB0E(Sat(\phi_1), Sat(\phi_2)),$$

$$S^{yes} = Sat(\phi_2),$$

$$S^{?} = S \setminus (S^{no} \cup S^{yes})$$

onde os algoritmos PROB0A, PROB1E e PROB0E são descritos a seguir.

Ao computar $p_s^{max}(\phi_1 \mathcal{U} \phi_2)$, S^{no} contém todos os estados para os quais $p_s^A(\phi_1 \mathcal{U} \phi_2) = 0$, para todo adversário A. Isso é determinado pela pré computação descrita pelo algoritmo PROBA0.

Em contrapartida, S^{yes} contém todos os estados para os quais $p_s^A(\phi_1 \mathcal{U} \phi_2) = 1$ para algum adverário A. Para isso, usamos o algoritmo PROB1E. Ambos são dados a seguir.

Algorithm 4: PROB0A($Sat(\phi_1), Sat(\phi_2)$)

begin
  $X := Sat(\phi_2)$
  $done := false$
  while *(*$done = false$*)* **do**
    $X' := X \cup \{s \in Sat(\phi_1) | \exists (a, \mu) \in R(s). \exists s' \in X. \mu(s') > 0\}$
    if $(X' = X)$ **then**
      $done := true$
    end
    $X := X'$
  end
  return $S \setminus X$
end

Algorithm 5: PROB1E($Sat(\phi_1, Sat(\phi_2))$

```
begin
    X := S
    done := false
    while (done = false) do
        X' := Sat(φ2)
        done' := false
        while (done' = false) do
            X'' := X' ∪ {s ∈ Sat(φ1) | ∃(a, μ) ∈ R(s).
                  (∀s' ∈ X.μ(s') > 0 → s' ∈ X) ∧ (∃s' ∈ X'.μ(s') > 0)}
            if (X'' = X') then
                done' := true
            end
            X' := X''
        end
        if (X' = X) then
            done := true
        end
        X := X'
    end
    return X
end
```

Algorithm 6: PROB0E($Sat(\phi_1), Sat(\phi_2)$)

```
begin
    X := Sat(φ2)
    done := false
    while (done = false) do
        X' := X ∪ {s ∈ Sat(φ1) | ∀(a, μ) ∈ R(s).∃s' ∈ X.μ(s') > 0}
        if (X' = X) then
            done := true
        end
        X := X'
    end
    return S \ X
end
```

PROB0A computa primeiro o estado onde é possível, sob o controle de um adversário e com probabilidade maior que zero, atingir um estado que satisfaça ϕ_2 sem sair de estados satisfazendo ϕ_1. Depois, estes estados são subtraídos de S para determinar os estados que possuem probabilidade zero para todo adversário.

PROB1E trabalha identificando estados para os quais $p_s^{max}(\phi_1 \mathcal{U} \phi_2)$ é menor que 1. Como baseia-se na computação de um duplo ponto fixo, é implementado com dois laços

aninhados. O mais externo computa o conjunto de estados X. Ao final do algoritmo X conterá todos os estados de S para os quais $p_s^A(\phi_1 \mathcal{U} \phi_2) = 1$ para algum adversário A. X é inicializado com S e a cada interação do laço mais externo, estados para os quais não há nenhum adversário A com $p_s^A(\phi_1 \mathcal{U} \phi_2) = 1$ são removidos de X. O laço mais interno remove os estados dos quais não se é possível alcançar um estado em $Sat(\phi_2)$ sem passar por um estado que não esteja em $Sat(\phi_1)$ ou por um estado que já tenha sido removido de X.

Ao computar $p_s^{min}(\phi_1 \mathcal{U} \phi_2)$, S^{no} contém todos os estados para os quais $p_s^A(\phi_1 \mathcal{U} \phi_2) = 0$ para algum adverário A. Isto é feito através do algoritmo PROB0E, que opera de forma semelhante a PROB0A.

A computação de $p_s^{max}(\phi_1 \mathcal{U}^{\leq k} \phi_2)$ e $p_s^{min}(\phi_1 \mathcal{U}^{\leq k} \phi_2)$ para os demais estados em $S^?$, conforme observou [Bai98], pode ser feita tirando proveito do fato de que $p_s^{max} = \lim_{n \to \infty} p_s^{max(n)}$, onde:

$$p_s^{max(n)} = \begin{cases} 1 & \text{se } s \in S^{yes} \\ 0 & \text{se } s \in S^{no} \\ 0 & \text{se } s \in S^? \text{ e } n = 0 \\ max_{(a,\mu) \in R(s)} \left\{ \sum_{s' \in S} \mu(s').p_s^{max(n-1)} \right\} & \text{se } s \in S^? \text{ e } n > 0 \end{cases}$$

e $p_s^{min} = \lim_{n \to \infty} p_s^{min(n)}$, onde:

$$p_s^{min(n)} = \begin{cases} 1 & \text{se } s \in S^{yes} \\ 0 & \text{se } s \in S^{no} \\ 0 & \text{se } s \in S^? \text{ e } n = 0 \\ min_{(a,\mu) \in R(s)} \left\{ \sum_{s' \in S} \mu(s').p_s^{min(n-1)} \right\} & \text{se } s \in S^? \text{ e } n > 0 \end{cases}$$

Os valores de $p_s^{max}(\phi_1 \mathcal{U} \phi_2)$ e $p_s^{min}(\phi_1 \mathcal{U} \phi_2)$ podem então ser aproximados através de métodos de computação interativa, calculando $p_s^{max(n)}$ e $p_s^{min(n)}$ para sucessivos valores de n, terminando quando algum critério de convergência é atingido. Assumindo que R é representado em uma matriz, como nos casos anteriores, cada interação deste método pode ser feita da mesma forma, através de uma multiplicação matriz-vetor e uma operação "máximo" ou "mínimo".

Algoritmo para o operador $\mathcal{E}_{\bowtie c}[\phi]$

Para o operador de custo esperado as computações de $e_s^{max}(\phi)$ e $e_s^{min}(\phi)$ são necessárias.

Computamos primeiro os conjuntos de estados S^0 e S^∞, cujo custo esperado máximo/mínimo de atingir um estado satisfazendo ϕ é respectivamente 0 e ∞. Em ambos os casos, o conjunto de estados S^0 é igual ao conjunto de estados que satisfaz ϕ. Ao computar e_s^{max}, S^∞ contém todos os estados para os quais $p_s^A(\diamond\phi) < 1$ para algum adversário A. Ao computar e_s^{min}, S^∞ contém todos os estados para os quais $p_s^A(\diamond\phi) < 1$ para todo adversário A. Assim sendo, podemos identificar S^∞ para os casos máximo e mínimo usando as técnicas da seção anterior para fazer a checagem de $\neg P_{\geq 1}[\diamond\phi]$ e $P_{<1}[\diamond\phi]$, respectivamente.

A computação de $e_s^{max}(\phi)$ e $e_s^{min}(\phi)$ para os demais estados em $S^? = S \setminus (S^0 \cup S^\infty)$ é similar a computação de $p_s^{max}(\phi_1 \mathcal{U} \phi_2)$ e $p_s^{min}(\phi_1 \mathcal{U} \phi_2)$, descritas na seção anterior, e pode ser feita através de um método interativo, computando $e_s^{max} = \lim_{n\to\infty} e_s^{max(n)}$, onde:

$$e_s^{max(n)} = \begin{cases} \infty & \text{se } s \in S^\infty \\ 0 & \text{se } s \in S^0 \\ 0 & \text{se } s \in S^? \text{ e } n = 0 \\ max_{(a,\mu)\in R(s)} \left\{ c(a) + \sum_{s' \in S^?} \mu(s').e_s^{max(n-1)} \right\} & \text{se } s \in S^? \text{ e } n > 0 \end{cases}$$

e $e_s^{min} = \lim_{n\to\infty} e_s^{min(n)}$, onde:

$$e_s^{min(n)} = \begin{cases} \infty & \text{se } s \in S^\infty \\ 0 & \text{se } s \in S^0 \\ 0 & \text{se } s \in S^? \text{ e } n = 0 \\ min_{(a,\mu)\in R(s)} \left\{ c(a) + \sum_{s' \in S^?} \mu(s').e_s^{min(n-1)} \right\} & \text{se } s \in S^? \text{ e } n > 0 \end{cases}$$

Os valores de $e_s^{max}(\phi)$ e $e_s^{min}(\phi)$ podem então ser aproximados interativamente, e o processo termina quando o nível de precisão desejado é atingido.

3.2.4 Complexidade do processo de verificação para PCTL sobre MDPs

A complexidade de tempo para a checagem de uma fórmula PCTL sobre um modelo MDP $\mathcal{M} = (S, s_0, R)$ é linear no tamanho da fórmula e polinomial no tamanho do modelo [RKNP04]. O tamanho de um modelo MDP é definido como $|\mathcal{M}| = \sum_{s\in S} |R(s)|$,

ou seja, o número total de possíveis escolhas não determinísticas. Como $R(s)$ é não vazio para todo s, é sempre o caso que $|\mathcal{M}| > S$. Leva-se em conta também que é necessário checar um operador de ϕ de cada vez. Os casos mais custosos são os dos operadores"until" e "custo esperado", para os quais é necessário resolver um problema de otimização linear, o que pode ser feito em tempo polinomial [RKNP04].

3.3 Modelagem e verificação de sistemas probabilísticos de tempo contínuo

Veremos a seguir outro modelo para sistemas probabilísticos, onde as transições são feitas em tempo real – Cadeias de Markov de Tempo Contínuo (CTMC). Esta classe de modelos também estende a classe das DTMCs, porém de outra forma: enquanto cada transição de estados nas DTMCs são feitas em passos discretos no tempo, nas CTMCs as transições podem ocorrer em tempo real.

3.3.1 Modelagem: Cadeias de Markov de Tempo Contínuo

Uma CTMC é uma família de variáveis randômicas $\{X(t)|t \geq 0\}$, onde $X(t)$ é uma observação feita no instante t e t varia sobre os números reais não negativos. O espaço de estados é discreto e corresponde ao conjunto de todos os valores que $X(t)$ pode assumir; assumiremos também para que o espaço de estados é finito, para o propósito desta tese. Uma CTMC deve satisfazer à propriedade de Markov, que aplicada a este modelo pode ser formulada da seguinte maneira: para todo inteiro $k \geq 0$, toda seqüência de instantes de tempo $t_0 \leq t_1 \leq \cdots \leq t_k$ e estados $s_0, \cdots, s_k$:

$$P[X(t_k) = s_k | X(t_{k-1}) = s_{k-1}, \cdots, X(t_0) = s_0] = P[X(t_k) = s_k | X(t_{k-1}) = s_{k-1}]$$

A intuição por trás da propriedade de Markov das CTMC é que a probabilidade de fazer uma transição para um determinado estado depende apenas do estado atual. Além disso, essa probabilidade é independente do tempo que o sistema passou no estado corrente até o presente momento, ou seja, a distribuição não tem memória sobre o seu comportamento até o momento presente. Essa é uma conseqüência natural da propriedade de Markov, conhecida como "propriedade da ausência de memória".

Há apenas uma distribuição contínua de probabilidades que exibe a propriedade descrita: a distribuição exponencial (também conhecida como distribuição exponencial negativa). Uma taxa λ é associada à transição entre dois estados, e usada como parâmetro da distribuição exponencial. A probabilidade de fazer uma transição entre o par de estados em t unidades de tempo é $1 - e^{-\lambda t}$. Intuitivamente, a taxa λ representa o número médio de vezes que a transição ocorreria por unidade de tempo.

Definição 3.3.1 *(Cadeia de Markov de Tempo Contínuo – CTMC)*
Uma CTMC é uma tupla $(S, s_0, \mathbf{R})$ *onde:*

- S *é um conjunto finito de estados;*
- $s_0 \in S$ *é o estado inicial;*
- $\mathbf{R} : S \times S \to \mathbb{R}_{\geq 0}$ *é a matriz de taxas de transição.*

A matriz de taxas de transição associa taxas a cada par de estados da CTMC. Uma transição entre os estados s e s' só pode ocorrer se $\mathbf{R}(s, s') > 0$. Tipicamente em um estado s existe mais de um estado s' para o qual $\mathbf{R}(s, s') > 0$. A primeira transição a ocorrer determina o próximo estado da CTMC. O tempo passado no estado s, antes que qualquer transição ocorra, é exponencialmente distribuído com taxa $E(s) = \sum_{s' \in S} \mathbf{R}(s, s')$. $E(s)$ é a *taxa de saída* do estado s.

Podemos também determinar a probabilidade de que um estado s' seja o próximo estado para o qual uma transição seja feita a partir de um estado s, independente do tempo em que a transição ocorra.

Definição 3.3.2 *A cadeia de Markov embutida emb(C) de uma CTMC* $\mathcal{C} = (S, s_0, \mathbf{R})$ *é uma Cadeia de Markov de Tempo Discreto (DTMC)* $(S, s_0, \mathbf{P})$ *onde para* $s, s' \in S$*:*

$$\mathbf{P}(s, s') = \begin{cases} \mathbf{R}(s, s')/E(s) & \textit{se } E(s) \neq 0 \\ 1 & \textit{se } E(s) = 0 \textit{ e } s = s' \\ 0 & \textit{caso contrário} \end{cases}$$

As definições anteriores permitem considerar o comportamento de uma CTMC de uma forma alternativa. Uma CTMC permanece em um estado s por um tempo que é exponencialmente distribuído com taxa $E(s)$ e então realiza uma transição. A probabilidade de que esta transição seja para o estado s' é dada por $\mathbf{P}(s, s')$. A matriz $\mathbf{Q}$ definida a seguir será usada para o processo de análise de verificação das CTMCs.

Definição 3.3.3 *A matriz geradora infinitesimal* $\mathbf{Q} : S \times S \to \mathbb{R}$ *é definida como:*

$$\mathbf{Q}(s, s') = \begin{cases} \mathbf{R}(s, s') & \textit{se } s \neq s' \\ -\sum_{s'' \neq s} \mathbf{R}(s, s'') & \textit{caso contrário} \end{cases}$$

Assim como fizemos com os modelos MDP, para analisar modelos CTMCs rotularemos a CTMC com informações adicionais. Utilizamos uma função de rotulação L, que mapeia os estados para conjuntos de proposições atômicas de um conjunto AP. Além disso, associamos dois tipo de custos ao modelo: *custos instantâneos* e *custos acumulativos*, definidos respectivamente pelas funções de custo $\mathbf{C}$ e $\mathbf{c}$. O custo instantâneo $\mathbf{C}(s, s')$ é computado cada vez que uma transição entre os estados s e s' ocorre. O custo acumulativo $\mathbf{c}(s)$ é a taxa que determina a o custo de permanência no estado s, isto é, um custo de $t \cdot c(s)$ é computado se a CTMC permanece no estado s por t unidades de tempo.

Definição 3.3.4 *Uma CTMC rotulada é uma tupla* $(\mathcal{C}, L, \mathbf{C}, \mathbf{c})$ *onde:*

- $\mathcal{C} = (S, s_0, \mathbf{R})$ *é uma CTMC;*
- $L : S \to 2^{AP}$ *é uma função de rotulação;*
- $\mathbf{C} : S \times S \to \mathbb{R}^{\geq 0}$ *é uma função de custo instantâneo;*
- $\mathbf{c} : S \to \mathbb{R}^{\geq 0}$*é uma função de custo acumulativo.*

A figura 3.2 mostra uma CTMC de três estados, onde 0 é o estado inicial. Esta CTMC, retirada de [RKNP04], modela uma fila de processos, onde o estado i indica que existem i processos na fila. Os processos chegam com taxa $1,5$ e são removidos da fila com taxa 3. O estado 0 está rotulado com a proposição $\{\text{vazio}\}$, e o estado 3 com a proposição $\{\text{cheio}\}$. A matriz de transição associada $\mathbf{R}$, a matriz geradora infinitesimal Q e a matriz de probabilidades da cadeia de Markov embutida são as seguintes:

$$\mathbf{R} = \begin{pmatrix} 0 & 1,5 & 0 & 0 \\ 3 & 0 & 1,5 & 0 \\ 0 & 3 & 0 & 1,5 \\ 0 & 0 & 3 & 0 \end{pmatrix},$$

$$\mathbf{Q} = \begin{pmatrix} -1,5 & 1,5 & 0 & 0 \\ 3 & -4,5 & 1,5 & 0 \\ 0 & 3 & -4,5 & 1,5 \\ 0 & 0 & 3 & -3 \end{pmatrix}, \mathbf{P} = \begin{pmatrix} 0 & 1 & 0 & 0 \\ \frac{2}{3} & 0 & \frac{1}{3} & 0 \\ 0 & \frac{2}{3} & 0 & \frac{1}{3} \\ 0 & 0 & 1 & 0 \end{pmatrix}.$$

Definição 3.3.5 *Um caminho em uma CTMC é uma seqüência não vazia* $s_0 t_0 s_1 t_1 s_2 \cdots$ *onde* $s_0, s_1, \cdots$ *são estados de* S, $\mathbf{R}(s_i, s_{i+1}) > 0$ *e* $t_i \in \mathbb{R}_{>0}$ *para todo* $i \geq 0$.

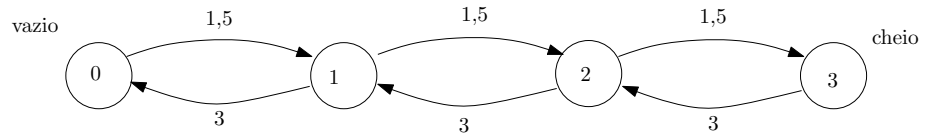


Figura 3.2: Uma CTMC de quatro estados.

O valor de t_i representa o tempo transcorrido no estado s_i. Assim como em MDPs, utilizaremos $w(i)$ para denotar o i-ésimo estado de um caminho w (que por sua vez denotaremos por s_i). time(w, i) será a notação para a quantidade de tempo passada no estado s_i, ou seja, t_i do caminho w. $w@t$ o estado ocupado no tempo t, isto é, $w(k)$ onde k é o menor índice para o qual $\sum_{i=0}^{k} t_i \geq t$.

Utilizaremos $Path_s$ para denotar o conjunto de todos os caminhos iniciados no estado s. A medida de probabilidade $Prob_s$ sobre $Path_s$ pode ser definida da seguinte forma [BKH99]: se os estados $s_0, \cdots s_n \in S$ satisfazem $\mathbf{R}(s_i, s_{i+1}) > 0$ para todo $0 \leq i < n$, e $I_0, \cdots, I_{n_1}$ são intervalos não vazios em $\mathbb{R}_{\geq 0}$, então o conjunto cilíndrico $C(s_0, I_0, \cdots, I_{n-1}, s_n)$ é definido como o conjunto infinito contendo todos os caminhos $s'_0 t_0 s'_1 t_1 s'_2 \cdots$ onde $s_i = s'_i$ para $i \leq n$ e $t_i \in I_i$ para $i < n$.

Seja $\sum_s$ a menor σ-algebra em $Path_s$, que contem todos os conjuntos cilíndricos $C(s_0, I_0, \cdots I_{n-1}, s_n)$, onde $s_0, \cdots s_n \in S$ varia sobre todas as seqüências de estados com $s_0 = s$ e $\mathbf{R}(s_i, s_{i+1}) > 0$ para $0 \leq i < n$, e $I_0, \cdots I_{n-1}$ variam sobre todas as seqüências de intervalos não vazios de $\mathbb{R}_{\geq 0}$.

Definição 3.3.6 *A medida de probabilidade $Prob_s$ sobre $\sum_s$ é a única medida definida indutivamente por $Prob_s(C(s_0)) = 1$ e $Prob_s(C(s_0, \cdots, s_n, I_n, s_{n+1}))$ é igual a:*

$$Prob_s(C(s_0, \cdots, s_n)) \cdot \mathbf{P}(s_n, s_{n+1}) \cdot (e^{-infI_n \cdot E(s_n)} - e^{-supI_n \cdot E(s_n)})$$

onde $\mathbf{P}$ é a matriz de probabilidades de transição da cadeia de Markov embutida na CTMC.

O seguinte exemplo foi apresentado em [RKNP04].

Exemplo 3.3.1 *Considerando a CTMC da figura 3.2 e a seqüência de estados e intervalos $0, [0, 2], 1$, (isto é, $s_0 = 0$, $I_0 = [0, 2]$ e $s_1 = 1$). Usando a medida de probabilidade para o estado inicial 0 da CTMC, temos para o conjunto cilíndrico $C(0, [0, 2], 1)$:*

$$Prob_0(C(0,[0,2],1)) = 1 \cdot \mathbf{P}(0,1) \cdot (e^{-0 \cdot E(0)} - e^{-2 \cdot E(0)}) = 1 \cdot 1 \cdot (e^0 - e^{-3}) = 1 - e^{-3}$$

.

Intuitivamente, isto significa que a probabilidade de deixar o estado inicial 0 *e passar ao estado* 1 *nas primeiras* 2 *unidades de tempo é* $1 - e^{-3} \simeq 0,950213$.

Usando esta medida de probabilidade podemos agora definir o custo esperado de alcançar um determinado conjunto de estados.

Definição 3.3.7 *Para um estado s, um conjunto de estados alvo $F \subseteq S$ e um caminho $w \in Path_s$, definimos o custo $cost(F)(w)$ de alcançar um estado em F ao longo do caminho w como:*

$$cost(F)(w) = \begin{cases} \sum_{i=1}^{min\{j|w(j)\in F\}} & \mathbf{C}(w(i-1), w(i)) + c(w(i-1)) \cdot time(w, i-1), \\ & \textit{se } \exists j \in \mathbb{N},\ w(j) \in F \\ \infty & \textit{caso contrário} \end{cases}$$

Sendo E_s a esperança com respeito a medida de probabilidade $Prob_s$, o custo esperado de atingir F a partir de s é $E_s(cost(F))$. O custo de um caminho que não atinge F é definido como ∞, então o custo esperado de atingir F a partir de s é finito se e somente se um estado em F é atingido a partir de s com probabilidade 1.

Além das probabilidades de caminho, o modelo de CTMC nos permite considerar outras propriedades: *comportamento transiente* e *comportamento de estacionário*. O comportamento transiente se refere ao estado do modelo em um instante de tempo em particular. O comportamento estacionário descreve o estado de uma CTMC após longo tempo de execução.

Definição 3.3.8 *A probabilidade transiente $\pi_{s,t}(s')$ é definida como a probabilidade, uma vez tendo começado no estado s, de estar no estado s' no tempo t.*

$$\pi_{s,t}(s') = Prob_s(\{w \in Path_s | w@t = s'\})$$

Definição 3.3.9 *A probabilidade do estado estacionário, também conhecida como probabilidade do equilíbrio estacionário, $\pi_s(s')$ é definida como a probabilidade, uma vez tendo começado no estado s, de estar no estado s' na longa jornada, ou seja, após a passagem de um tempo considerável.*

$$\pi_s(s') = \lim_{t \to \infty} \pi_{s,t}(s')$$

A distribuição de probabilidades do estado estacionário, ou seja, os valores $\pi_s(s')$ para todo $s \in S$, pode ser usada para inferir a porcentagem do tempo, na longa jornada, que a CTMC passa em cada estado. Para a classe de CTMCs de interesse deste trabalho, que são as CTMCs homogêneas e de estado finito, o limite da definição acima sempre existe [Ste94]. Além disso, para CTMCs que são irredutíveis (aquelas para as quais existe um caminho finito entre cada par de estados), as probabilidades de estado estacionário $\pi_s(s')$ são independentes do estado inicial s.

3.3.2 Especificação: Lógica estocástica de tempo contínuo CSL

Apresentaremos uma extensão da lógica PCTL (descrita na seção 3.2.2) para escrever especificações de CTMCs: Lógica estocástica de tempo contínuo (CSL). A lógica CSL foi introduzida primeiramente em [ASSB96], e depois estendida em [BKH99]. CSL permite raciocinar sobre propriedades de caminhos, comportamento transiente e comportamento estacionário.

Definição 3.3.10 *(Sintaxe de fórmulas CSL)*

A sintaxe de fórmulas de CSL é:

$\phi ::= true \mid a \mid \neg\phi \mid \phi \wedge \phi \mid \mathcal{P}_{\bowtie p}[\psi] \mid \mathcal{S}_{\bowtie p}[\phi] \mid \mathcal{E}_{\bowtie c}[\phi]$, $\psi ::= \phi \mid \phi \mathcal{U}^I \phi \mid \phi \mathcal{U} \phi$

onde a é uma proposição atômica, $\bowtie \in \{\leq, <, \geq, >\}$, $p \in [0,1]$, $c \in \mathbb{R}^{\geq 0}$ e I é um intervalo de $\mathbb{R}^{\geq 0}$

Assim como para PCTL, $\mathcal{P}_{\bowtie p}[\psi]$ indica que a probabilidade de uma fórmula de caminho ψ ser satisfeita a partir de um certo estado respeita o limite $\bowtie p$. As fórmulas de caminho de CSL são as mesmas de PCTL, exceto pelo parâmetro da versão limitada do

operador $\mathcal{U}$, que para este caso é um intervalo dos reais não negativos, ao invés de um limite inteiro. Uma fórmula de caminho $\phi_1 \mathcal{U}^I \phi_2$ vale se ϕ_2 é satisfeita em algum instante de tempo no intervalo I e ϕ_1 vale em todos os instantes anteriores.

O operador $\mathcal{S}$ descreve o comportamento estacionário da CTMC. A fórmula $\mathcal{S}_{\bowtie p}[\phi]$ representa que a probabilidade de estado estacionário de estar em um estado que satisfaz ϕ respeita o limite $\bowtie p$. Assim como definido para PCTL, a fórmula $\mathcal{E}_{\bowtie c}[\phi]$ representa que o custo esperado de atingir um estado satisfazendo ϕ respeita o limite $\bowtie c$. A diferença é que neste caso o custo é uma combinação dos custos instantâneos e acumulativos.

Escrevemos $s \models \phi$ para denotar que uma fórmula de CSL é satisfeita em um estado s de uma CTMC, e denotamos por $Sat(\phi)$ o conjunto $\{s \in S | s \models \phi\}$. Para uma fórmula de caminho ψ satisfeita por um caminho w, escrevemos $w \models \psi$. A semântica de CSL sobre modelos CTMC é dada a seguir.

Definição 3.3.11 *(Relação de Satisfação para fórmulas de CSL)*

Seja $\mathcal{C} = (S, S_0, \mathbf{R}, L, \mathbf{C}, \mathbf{c})$ *uma CMTC rotulada . A relação de satisfação para fórmulas CSL é definida da seguinte forma:*

$$
\begin{array}{lll}
s \models true & \textit{para todo} & s \in S, \\
s \models a & \textit{se e somente se} & a \in L(s), \\
s \models \neg\phi & \textit{se e somente se} & s \not\models \phi, \\
s \models \phi_1 \wedge \phi_2 & \textit{se e somente se} & s \models \phi_1 \textit{ e } s \models \phi_2, \\
s \models \mathcal{P}_{\bowtie p}[\psi] & \textit{se e somente se} & p_s(\psi) \bowtie p, \\
s \models \mathcal{S}_{\bowtie p}[\phi] & \textit{se e somente se} & \sum_{s' \models \phi} \pi_s(s') \bowtie p, \\
s \models \mathcal{E}_{\bowtie c}[\phi] & \textit{se e somente se} & e_s(\phi) \bowtie c\,,
\end{array}
$$

onde:

$p_s(\psi) \stackrel{def}{=} Prob_s(\{w \in Path_s | w \models \psi\})$,

$e_s(\phi) \stackrel{def}{=} E_s(cost(\{t \in S | t \models \phi\})$

e para cada caminho $w \in Path_s$*:*

$$
\begin{array}{lll}
w \models \phi & \textit{se e somente se} & w(1) \models \phi, \\
w \models \phi_1 \mathcal{U}^I \phi_2 & \textit{se e somente se} & \exists t \in I \textit{ tal que } w@t \models \phi_2 \wedge w@x \models \phi_1 \forall x \in [0, t), \\
w \models \phi_1 \mathcal{U} \phi_2 & \textit{se e somente se} & \exists k \geq 0 \textit{ tal que } w(k) \models \phi_2 \wedge w(j) \models \phi_1 \forall j < k.
\end{array}
$$

Como discutido em [BKH99], para qualquer fórmula de caminho ψ, o conjunto $\{w \in Path_s | w \models \psi\}$ é um conjunto mensurável de $Path_s$, de forma que $Prob_s$ é bem definida sobre este conjunto.

Assim como em PCTL, os operadores $false$, $\vee$ e $\rightarrow$ podem ser definidos para CSL em função de $true$, $\neg$ e $\wedge$. Podemos também definir operadores de temporais $\diamond$ e $\Box$:

$$\mathcal{P}_{\bowtie p}[\diamond\phi] \equiv \mathcal{P}_{\bowtie p}[true\mathcal{U}\phi], \qquad \mathcal{P}_{\bowtie p}[\diamond^I\phi] \equiv \mathcal{P}_{\bowtie p}[true\mathcal{U}^I\phi]$$

$$\mathcal{P}_{\bowtie p}[\Box\phi] \equiv \mathcal{P}_{\overline{\bowtie} 1-p}[\diamond\neg\phi], \qquad \mathcal{P}_{\bowtie p}[\Box^I\phi] \equiv \mathcal{P}_{\overline{\bowtie} 1-p}[\diamond^I\neg\phi].$$

Apesar de que CSL não inclui explicitamente um operador para raciocinar sobre comportamento transiente, é possível raciocinar sobre a probabilidade de satisfazer uma fórmula ϕ em um dado instante t utilizando: $\mathcal{P}_{\bowtie p}[\diamond^{[t,t]}\phi]$.

Quando o operador mais externo de uma fórmula de CSL é $\mathcal{P}_{\bowtie p}$, $\mathcal{S}_{\bowtie p}$ ou $\mathcal{E}_{\bowtie c}$ podemos omitir o limite ($\bowtie p$ ou $\bowtie c$) e retornar o valor real da probabilidade associada ou do custo esperado.

3.3.3 Verificação de especificações em CSL sobre modelos CTMC

Conforme mostrado em [ASSB96], o processo de verificação para CSL é decidível para limites de tempo racionais. Algoritmos foram apresentados em [BKH99]. Descreveremos a seguir os algoritmos tal como mostrados em [RKNP04], que são os de [BKH99] com as melhorias de [BHHK00] e [KKNP01]. Não descreveremos os algoritmos em detalhe porque, a menos de recursos matemáticos, as idéias por detrás dos procedimentos são similares às apresentadas para verificação de PCTL (seção 3.2.3). Discutiremos somente o algoritmos para tratamento dos operadores $\mathcal{P}_{\bowtie p}$, $\mathcal{P}_{\bowtie p}[\mathcal{U}]$, $\mathcal{P}_{\bowtie p}[\mathcal{U}^I]$, $\mathcal{S}_{\bowtie p}$ e $\mathcal{E}_{\bowtie p}$, uma vez que os outros operadores podem ser tratados com os mesmo algoritmos apresentados na seção 3.1.3.

Algoritmo para os operadores $\mathcal{P}_{\bowtie p}[\phi]$ e $\mathcal{P}_{\bowtie p}[\phi_1\mathcal{U}\phi_2]$

A verificação destes operadores não está relacionada com os aspectos de tempo real da CTMC, e portanto o processo é semelhante ao que foi mostrado para os MDPs, considerando a distribuição dada pela DTMC embutida na CTMC.

Algoritmo para o operador $\mathcal{P}_{\bowtie p}[\phi_1 \mathcal{U}^I \phi_2]$

Para este operador necessitamos determinar as probabilidades $p_s(\phi_1 \mathcal{U}^I \phi_2)$ para todos os estados s onde I é um intervalo arbitrário de números reais não negativos. Observando que $p_s(\phi_1 \mathcal{U}^I \phi_2) = p_s(\phi_1 \mathcal{U}^{cl(I)} \phi_2)$, onde $cl(I)$ é o fecho do intervalo I, e que $p_s(\phi_1 \mathcal{U}^{[0,\infty)} \phi_2) = p_s(\phi_1 \mathcal{U} \phi_2)$, precisamos apenas tratar os seguintes casos para o intervalo I:

- $I = [0, 1]$ para $t \geq 0$;
- $I = [t, t']$ para $t > 0$ e $t \geq t'$;
- $I = [t, \infty)$ para $t > 0$.

Estas probabilidades podem ser calculadas utilizando variantes das técnicas padrão para análise transiente, conforme os métodos de cálculo apresentados em [BHHK00] e [KKNP01].

Algoritmo para o operador $\mathcal{S}_{\bowtie p}[\phi]$

Um estado s satisfaz a fórmula $\mathcal{S}_{\bowtie p}[\phi]$ de estado estacionário se $\sum_{s' \models \phi} \pi_s(s') \bowtie p$. Assim, para verificar o operador $\mathcal{S}_{\bowtie p}[\phi]$, é necessário computar as propriedades de estado estacionário $\pi_s(s')$ para todos os estados s e s'. Para CTMCs irredutíveis, estas probabilidades são independentes do estado inicial s; neste caso, denotamos a propriedade do estado estacionário de estar no estado s' por $\pi(s')$ e o vetor de todos os respectivos valores para $s' \in S$ como $\underline{\pi}$. $\underline{\pi}$ pode ser computado resolvendo o sistema de equações lineares:

$$\underline{\pi} \cdot \mathbf{Q} = \underline{0} \text{ e } \sum_{s' \in S} \pi(s') = 1.$$

A satisfação da fórmula CSL, que neste caso será a mesma para todos os estados $s \in S$ da CTMC, pode ser determinada somando as probabilidades de estado estacionário dos estados satisfazendo ϕ e comparando com o limite na fórmula:

$$s \models \mathcal{S}_{\bowtie p}[\phi] \Leftrightarrow \sum_{s' \models \phi} \pi(s') \bowtie p.$$

Para o caso em que a CTMC não é irredutível, o procedimento é mais complexo. Definimos uma *componente fortemente conexa* – CFF de uma CTMC como um conjunto de estados para os quais existe um caminho finito entre qualquer par de estados. Definimos uma *componente fortemente conexa de base* – BCFF como uma componente fortemente conexa tal que não existem transições na CTMC para estados fora da componente. Cada BCFF individual B pode ser tratada como uma CTMC irredutível, para a qual podemos calcular as probabilidades de estado estacionário tal como descrito anteriormente. Estas probabilidades são independentes do estado inicial e as denominaremos $\pi^B(s')$ para $s' \in B$. No caso especial onde B compreende um único estado s', $\pi^B(s') = 1$.

A seguir, determinamos a probabilidade de alcançar cada BCFF B do estado s. Isto é simplesmente $p_s(\diamond a_B)$, onde a_B é uma proposição atômica verdadeira apenas nos estados $B \subseteq S$. Podemos computar $p_s(\diamond a_B)$ da mesma forma que seria feito para verificar a fórmula $\mathcal{P}_{\bowtie p}[\diamond a_B] \equiv \mathcal{P}_{\bowtie p}[true \mathcal{U} a_B]$, o que se reduz a solução de um sistema de equações lineares. Então, para os estados s, $s' \in S$:

$$\pi_s(s') = \begin{cases} p_s(\diamond a_B) \cdot \pi^B(s') & \text{se } s' \text{ está em uma BCFF } B \\ 0 & \text{caso contrário.} \end{cases}$$

Uma vez que as probabilidades de estado estacionário $\pi^B(s')$ são independentes de s e que o procedimento para calcular $p_s(\diamond a_B)$ o faz para todo $s \in S$ de uma só vez, o trabalho total requerido para computar $\pi_s(s')$ para todos $s, s' \in S$ é a solução de dois sistemas de equações lineares para cada BCFF na CTMC. A computação das BCFF em uma CTMC requer a análise de sua estrutura de grafo e pode ser realizada utilizando os algoritmo básicos baseados em busca em profundidade.

Algoritmo para o operador $\mathcal{E}_{\bowtie c}[\phi]$

Para este caso, computamos o custo esperado $e_s(\phi)$ para todos os estados $s \in S$. Podemos usar a DTMC embutida na CTMC. Para a DTMC embutida $emb(\mathcal{C}) = (S, s_0, \mathbf{P})$ da CTMC, identificamos os conjuntos de estados para os quais $e_s(\phi)$ é 0 ou ∞. Denotaremos estes conjuntos por S^0 e S^∞. O conjunto S^0 é o conjunto dos estados satisfazendo ϕ, enquanto S^∞ corresponde aos estados a partir dos quais a probabilidade de atingir um estado onde ϕ vale é menor que 1. Resta ainda computar $e_s(\phi)$ para os estados em $S^? = S \setminus (S^0 \cup S^\infty)$. Isto é feito computando o sistema de equações lineares sobre as

variáveis $\{x_s | s \in S^?\}$:

$$x_s = \left(\sum_{s' \in S^?} \mathbf{P}(s, s') \cdot (\mathbf{C}(s, s') + x_{s'}) \right) + \frac{1}{E(s)} \cdot \mathbf{c}(s)$$

e depois fazendo $e_s(\phi) = x_s$ para $s \in S^?$. Para todo $s \in S$, $E(s)$ denota a taxa de saída do estado s e então $\frac{1}{E(s)}$ é o tempo esperado que o sistema permaneça em s antes que uma transição ocorra. Isto pode ser calculado utilizando qualquer método direto ou interativo.

3.3.4 Complexidade do processo de verificação de especificações CSL sobre modelos CTMC

A complexidade de tempo para verificar uma fórmula ϕ de CSL sobre uma CTMC $\mathcal{C} = (S, s_0, \mathbf{R})$ é linear em $|\phi|$, polinomial em $|S|$ e linear em $q \cdot t_{max}$, onde $q = max_{s \in S} |\mathbf{Q}(s, s)|$ e t_{max} é o valor máximo encontrado em um parâmetro de um operador $\mathcal{U}$ limitado. Para $\mathcal{P}_{\bowtie p}[\phi_1 \mathcal{U} \phi_2]$ e $\mathcal{E}_{\bowtie c}[\phi]$, são requeridas soluções de sistemas de equações lineares de tamanho $|S|$. Isto pode ser feito empregando o método de eliminação Gaussiana, cuja complexidade é cúbica no tamanho do sistema. Para $\mathcal{P}_{\bowtie p}[\phi_1 \mathcal{U}^I \phi_2]$ é necessário calcular ao menos dois somatórios interativos, nos quais cada passo requer uma multiplicação vetor–matriz. Esta operação é quadrática no tamanho da matriz, ou seja, de $|S|$. O número total de interações necessárias é determinado pelo limite superior dado pelo algoritmo de Fox e Glynn [FG88], que para $q \cdot t$ grande é linear em $q \cdot t$.

Capítulo 4

Modelagem e verificação de propriedades epistêmicas em sistemas multi-agentes

Para atingir nosso objetivo de desenvolver modelos, linguagens e processos de verificação formal que permitam analisar propriedades relativas ao conhecimento de um agente ou grupo de agentes em um sistema concorrente, analisaremos três modelos para representação de sistemas concorrentes, discutindo para cada um deles adequações para a representação e verificação de propriedades envolvendo conhecimento. São discutidas também as características de cada modelo, explicitando as relações com as características temporais e epistêmicas (incertezas inerentes ao modelo) do sistema a ser modelado, e também os métodos apresentados para modelagem de conhecimento de forma coerente.

Tomamos a visão de que a evolução de um sistema pode ser o resultado da interação entre diferentes partes (ou agentes) do sistema. Assim, é necessária uma forma de especificar as partes, bem como mecanismos para combiná-las, possibilitando que modelos complexos sejam construídos de uma forma composicional.

O primeiro passo em direção ao nosso objetivo é propor um modelo composicional para verificação de modelos a nível de conhecimento e tempo em sistemas concorrentes, onde as relações de conhecimento sejam construídas diretamente a partir dos modelos locais de cada agente. Partimos do caso mais simples que já foi estudado em vários trabalhos correlatos: sistemas não probabilísticos e de tempo discreto; apresentamos para esta classe de sistemas uma abordagem composicional para obtenção de modelos globais e construção das relações de possibilidade para representar conhecimento. O segundo

passo é a apresentação de uma linguagem adequada para expressar propriedades epistêmicas, condizendo com as informações representadas no modelo; propomos uma linguagem para raciocinar sobre tempo e conhecimento, que resultou bastante semelhante a linguagens descritas em trabalhos correlatos contemporâneos a [BDR$^+$06], trabalho onde apresentamos a nossa proposta. O terceiro passo é a apresentação de algoritmos de verificação correspondentes; propomos um algoritmo com forte apelo intuitivo, cuja análise de complexidade mostra não aumentar a complexidade do processo convencional de verificação de modelos para linguagem não envolvendo conhecimento.

Após o ensaio com modelos não probabilísticos e de tempo discreto, investigamos como estender os resultados obtidos para outras classes de modelos, especificamente modelos envolvendo probabilidade e não determinismo. Os modelos baseados em cadeias de Markov foram considerados os mais adequados para embasarmos nossa proposta, uma vez que esta classe de modelos permitem modelar uma ampla gama de sistemas, ainda que permita processos de análise e verificação automática relativamente simples para os quais muitos estudos já foram feitos e outros estão em andamento, indicando que investimentos na área são promissores. Apresentamos modelos probabilísticos composicionais para modelar sistemas multi-agentes, e discutimos como as características temporais (tempo discreto ou contínuo) e de incerteza (não determinismo, probabilidade) influenciam o conhecimento dos agentes do sistema. Dois modelos são apresentados, para contemplar respectivamente sistemas probabilísticos e não determinísticos de tempo discreto, e sistemas não probabilísticos e não determinísticos de tempo contínuo.

Discutimos como adaptar as linguagens presentes na atual literatura para raciocinar sobre conhecimento e construir especificações envolvendo as propriedades de interesse características de cada classe de modelos abordada. Propomos algoritmos para verificação automática de especificações envolvendo conhecimento expressas nas linguagens adaptadas, sobre os modelos propostos.

4.1 Verificação Formal de propriedades epistêmicas em Sistemas não Probabilísticos Discretos

Apresentamos a seguir uma abordagem para verificação de propriedades epistêmicas em modelos para sistemas multi-agentes, conforme publicada em [BDR+06]. O trabalho foi embasado em resultados recentes em verificação de modelos e lógicas temporais de conhecimento, e foca em um mecanismo simples para se obter as relações de acessibilidade para conhecimento diretamente de especificações de sistemas baseadas em autômatos. Uma linguagem formal com semântica composicional é apresentada, juntamente com os respectivos algoritmos de verificação de modelos para verificação de propriedades epistêmicas.

Discutimos uma forma para obter uma especificação global completamente a partir de especificações locais para cada agente, incluindo as relações epistêmicas de acessibilidade e a função de valoração das proposições atômicas.

4.1.1 Modelagem de sistemas concorrentes não determinísticos

Sistemas multi-agentes, também chamados sistemas concorrentes, envolvem uma série de componentes que executam em conjunto. Normalmente as componentes têm meios para comunicarem-se, o que torna a computação de um sistema concorrente mais interessante do que o simples paralelismo de processos. Abordaremos nesta seção sistemas multi-agentes assíncronos, não determinísticos, onde o tempo é considerado discreto (não contínuo). Essa abordagem pode parecer contra intuitiva a princípio, porém esta é uma abstração comum em sistemas computacionais, que executam em máquinas que operam baseadas em "passos" e o tempo é contado em pulsos.

Conforme discutido na secção 3.1.1, a granularidade do modelo desejado depende do tipo de propriedades que se deseja verificar. Estamos interessados em verificar propriedades temporais e epistêmicas:

- Para modelar e verificar propriedades epistêmicas é necessário que o modelo capture o estado local individual de cada agente, além do estado global do sistema a cada ponto. Este é um requisito natural para que se possa identificar no modelo os estados indistinguíveis sob o ponto de vista de cada agente.

- Para modelar e verificar propriedades temporais, o modelo deve conter informação sobre os eventos e ações capazes de alterar o estado do sistema, e as possíveis seqüencias intercaladas de eventos e estados que podem acontecer durante todas as possíveis execuções não determinísticas do sistema.

De forma geral, o modelo deve capturar as interações e informações de cada componente e também o comportamento conjunto de todas as componentes, que dita o comportamento do sistema.

Nos referimos a [LMWF94] para uma abstração que represente sistemas concorrentes. O estilo de modelo proposto é operacional (ao invés de axiomático), considerado mais adequado para verificação automática através de checagem de modelos, e seguindo a linha apresentada para sistemas reativos, baseia-se em autômatos finitos.

Neste modelo, representamos o comportamento de cada agente (cada programa no protocolo P) como um autômato. Depois, é feita a composição dos autômatos correspondentes a todo o grupo de agentes (que fazem parte do sistema), identificando suas ações através de interfaces. Assim o modelo para o sistema concorrente é construído a partir de componentes previamente modeladas. O método de composição gera um único autômato que será o modelo para o sistema completo – cujo comportamento corresponde à execução paralela, porém monitorada, de todos os agentes.

Cada autômato i representa os estados locais (em seu conjunto de nós) e eventos do conjunto E (de arestas) para um agente.

Para adequar o modelo de autômatos à representação de sistemas concorrentes, propomos a construção de um modelo global a partir dos modelos de Kripke individuais dos agentes. Por simplificação, os modelos particulares dos agentes serão chamados de modelos locais, e o modelo do sistema concorrente - modelo global.

O primeiro passo da adequação do modelo local é enriquecer os autômatos com conjuntos de proposições atômicas. O autômato de cada agente local possui um conjunto particular indexado de proposições $\{p_j\}_{j\in\mathcal{I}}$, onde $\mathcal{I}$ é um conjunto de índices para o autômato i, que denota as propriedades relevantes sobre a computação realizada por i.

Vale ressaltar que o autômato para um agente representa o comportamento particular do agente, de forma que toda a informação relevante para o agente deve ser representada. O motivo para a indexação do conjunto de proposições é evitar conflitos do tipo dois

autômatos distintos referenciarem a mesma proposição com significados diferentes, o que acarretaria problemas ao compor o autômato do sistema.

As ações de um autômato serão classificadas como entrada (que representam eventos ocasionados pelo ambiente, como o recebimento de uma mensagem), saída (que o agente pode realizar e que afetam o ambiente, como o envio de uma mensagem) e internas (que o agente pode realizar, porém não são detectáveis pelo ambiente exceto através de seus efeitos em eventos de saída futuros, como a alteração do valor de uma variável local). O autômato gera saídas e ações internas de forma autônoma, e transmite ações de saída instantaneamente para o ambiente. As entradas do autômato são geradas pelo ambiente e transmitidas instantaneamente para o autômato. Um autômato é incapaz de bloquear uma ação de entrada. Para descrever formalmente esta classificação, cada autômato possui uma assinatura de suas ações.

Definição 4.1.1 *(Assinatura de ações)*

Uma assinatura de ações $\mathfrak{S}$ é um trio composto por três conjuntos de ações distintos dois a dois, chamados $in(\mathfrak{S}), out(\mathfrak{S})$, e $int(\mathfrak{S})$. As ações nesses conjuntos são chamadas respectivamente de ações de entrada, ações de saída, e ações internas de $\mathfrak{S}$. É também definido $ext(\mathfrak{S}) = in(\mathfrak{S}) \cup out(\mathfrak{S})$ como as ações externas de $\mathfrak{S}$, $local(\mathfrak{S}) = int(\mathfrak{S}) \cup out(\mathfrak{S})$ como as ações localmente controladas de $\mathfrak{S}$, e $acts(\mathfrak{S}) = in(\mathfrak{S}) \cup out(\mathfrak{S}) \cup int(\mathfrak{S})$ como as ações de $\mathfrak{S}$.

Definição 4.1.2 *(Modelo Local de Kripke)*

Seja $\mathfrak{S}$ uma assinatura de ações, e $\mathcal{I}$ um conjunto contável. Um modelo local de Kripke é uma estrutura $\mathfrak{A} = \langle \mathfrak{S}, S, S_0, E, \{p_j\}_{j\in\mathcal{I}}, \mathcal{L}\rangle$ tal que S é um conjunto de estados, $S_0 \subseteq S$ é um conjunto não vazio de estados iniciais, $E \subseteq S \times acts(\mathfrak{S}) \times S$ é um conjunto de arestas de transição, tal que para todo estado s' e ação de entrada π existe uma transição (s', π, s) em E. $\{p_i\}_{i\in\mathcal{I}}$ é um conjunto de proposições, e $\mathcal{L} : S \to 2^{\{p_j\}_{j\in\mathcal{I}}}$ a função de valoração que associa cada estado a um subconjunto de proposições.

Para obter o modelo global que represente o sistema resultante da execução concorrente do conjunto de agentes locais faremos a composição paralela dos modelos locais de Kripke para cada agente.

Definir a composição de assinaturas de ações é um passo preliminar à definição da composição dos modelos locais de Kripke. A composição das assinaturas de ações dos modelos locais resulta na assinatura de ações para o sistema concorrente. Para que a composição seja possível é necessário que os modelos locais satisfaçam algumas condições de compatibilidade.

Definição 4.1.3 *(Conjunto Compatível de Assinaturas de Ações)*

Seja I um conjunto de índices ao menos contável. Uma coleção $\{\mathfrak{S}_i\}_{i\in I}$ de assinaturas de ações é dita compatível se:

1. $out(\mathfrak{S}_i) \cap out(\mathfrak{S}_j) = \emptyset$*, para todo* $i, j \in I, i \neq j$
2. $int(\mathfrak{S}_i) \cap acts(\mathfrak{S}_j) = \emptyset$*, para todo* $i, j \in I, i \neq j$ *e*
3. *nenhuma ação está em* $acts(\mathfrak{S}_i)$ *para infinitos* i*.*

Estas restrições irão assegurar que uma ação não fique sob o controle de mais de um agente do sistema, e que ações internas de um agente não sejam detectáveis por outro, quando a composição dos modelos for realizada.

Definição 4.1.4 *(Composição de Assinaturas de Ações)*

A composição $\mathfrak{S} = \Pi_{i\in I}\mathfrak{S}_i$ de uma coleção de assinaturas de ações compatíveis $\{\mathfrak{S}_i\}_{i\in i}$ é a assinatura de ações onde:

- $in(\mathfrak{S}) = \cup_{i\in I}\, in(\mathfrak{S}_i) - \cup_{i\in I}\, out(\mathfrak{S}_i)$
- $out(\mathfrak{S}) = \cup_{i\in I}\, out(\mathfrak{S}_i)$
- $int(\mathfrak{S}) = \cup_{i\in I}\, int(\mathfrak{S}_i)$

Desta forma, ações de saída são aquelas que são saída de qualquer uma das assinaturas componentes, e o mesmo ocorre para ações internas. Ações de entrada são as ações que são ações de entrada para qualquer uma das componentes, não sendo ações de saída para nenhuma das assinaturas componentes. Vale ressaltar que interações entre as componentes são consideradas ações de saída da composição.

Agora podemos definir a composição dos modelos locais de Kripke. A idéia é que os estados do modelo composto (modelo global) sejam n-uplas cujas componentes correspondam a um estado local de cada agente, e as arestas de transição correspondam a todas as arestas presentes nos modelos locais. A operação de composição liga ações de saída de um modelo local com ações de entrada de mesmo nome de um número qualquer de outros modelos locais.

Definição 4.1.5 *(Composição paralela de modelos locais de Kripke: Modelo Global de Kripke)*

Uma coleção $\{\mathfrak{A}_i = \langle \mathfrak{S}_i, S_i, (S_0)_i, E_i, \{p_j\}_{j\in\mathcal{I}_i}, \mathcal{L}_i\rangle\}_{i\in I}$ *de modelos locais de Kripke é dita compatível se suas assinaturas de ações são compatíveis. A composição* $\mathfrak{A} = \Pi_{i\in I}\mathfrak{A}_i$ *de uma coleção compatível finita de modelos locais de Kripke* $\{\mathfrak{A}_i\}_{i\in I}$*, possui as seguintes componentes:*

- $\mathfrak{S} = \Pi_{i\in I}\ \mathfrak{S}_i$*,*
- $S = \Pi_{i\in I} S_i$*,*
- $S_0 = \Pi_{i\in I}(S_0)_i$*, e*
- $E \subseteq S \times acts(\mathfrak{S}) \times S$ *é o conjunto de ternos* $(s\prime, \pi, s)$ *tais que para todo* $i \in I$*,*
  - *se* $\pi \in \mathfrak{S}_i$*, então* $(s\prime[i], \pi, s[i]) \in E_i$*, e*
  - *se* $\pi \notin \mathfrak{S}_i$*, então* $s\prime[i] = s[i]$*,*

  onde a notação $[i]$ *representa a* i*-ésima componente do vetor de estados* s*.*
- $\{p_j\}_{j\in\mathcal{I},\mathcal{I}\in\ \{\mathcal{I}_i\}_{i\in I}} = \bigcup\{p_j\}_{j\in\mathcal{I}_i}$
- $\mathcal{L}$ *é uma função que associa a cada estado global um subconjunto de proposições, de acordo com as funções locais previamente definidas para cada agente:* $(s, p_j), j \in \mathcal{I}$ *está em* $\mathcal{L}$ *se e somente se* $j \in \mathcal{I}_i$ *e* $(s[i], p_j)$ *está em* $\mathcal{L}_i$*.*

Por simplificação, chamaremos tal composição de "Modelo Global de Kripke".

Estados no modelo global correspondem aos estados globais do sistema concorrente, e são os elementos no conjunto de estados globais C.

Um caminho em um modelo composto pode ser definido de forma análoga à definição apresentada para caminhos em modelos Kripke:

Definição 4.1.6 *(Caminho ou Execução em um modelo global de Kripke)*

Seja $\mathfrak{A} = \langle \mathfrak{S}, S, (S_0), E, \{p_j\}_{j \in \mathcal{I}, \mathcal{I} \in \{\mathcal{I}_i\}_{i \in I}}, \mathcal{L} \rangle$ *um modelo global de Kripke. Um caminho* π *em* $\mathfrak{A}$ *se caracteriza por uma seqüencia infinita de estados,* $\pi = s_0, s_1, s_2, \cdots$*, tal que para todo* $i \geq 0$*,* $(s_i, \pi, s_{i+1}) \in E$*, para alguma ação* π*.*

Ao fazer a composição paralela de um conjunto de autômatos onde cada autômato rege o comportamento de um agente, é possível e razoável obter um autômato global composto $\mathfrak{A} = \langle \mathfrak{S}, S, S_0, E, \{p_i\}_{i \in \mathcal{I}}, \mathcal{L} \rangle$ onde muitos estados de S correspondem a uma mesma componente de estado local para um agente em particular. Quando dois estados deste tipo s e t, cuja componente correspondente ao agente i represente o mesmo estado local para este agente, isto significa que o agente i é incapaz de perceber que houve uma mudança do estado global do sistema. Uma relação de equivalência para cada agente do sistema pode ser definida sobre o conjunto de estados globais com a propriedade de serem "indistinguíveis" sob o ponto de vista de um agente.

Definição 4.1.7 *(Relação de acessibilidade* $\sim_i$ *para o agente* i*)*

Seja $\mathfrak{A} = \langle \mathfrak{S}, S, S_0, E, \mathcal{P}, \mathcal{L} \rangle$ *a composição de um conjunto compatível de autômatos* $\{\mathfrak{A}_i\}_{i \in I} = \langle \mathfrak{S}_i, S_i, (S_0)_i, E_i, \{p_i\}_{i \in \mathcal{I}}, \mathcal{L}_i \rangle$ *para todo* $i \in [1, \ldots, n]$*. A relação de acessibilidade* $\sim_i \in S \times S$ *para cada agente* i *é a menor relação de equivalência contendo todos os pares* $(s\prime, s)$*,* $s\prime, s \in S$ *tais que* $s\prime[i] = s[i]$*.*

4.1.2 Especificação: lógica para conhecimento e tempo em sistemas multi-agentes KCTL

Descrevemos agora uma lógica para raciocinar sobre conhecimento e tempo em sistemas de transição de estados como o apresentado na seção 4.1.1. A linguagem utilizada é uma extensão de CTL com o operador modal de conhecimento (K_i). Chamaremos esta linguagem de KCTL.

KCTL é útil para expressar propriedades sobre conhecimento, tempo e eventos como por exemplo "O Agente A sabe que se mandar uma mensagem, o agente B eventualmente saberá a mensagem"(em KCTL: $K_A($ `msgsent` $\rightarrow \exists \mathsf{F} K_B$ `msg` $)$, considerando que a proposição `msgsent` significa "agente A enviou a mensagem `msg` para agente B" e a proposição `msg` representa o conteúdo da mensagem).

Assim como em CTL, fórmulas de KCTL raciocinam sobre propriedades de execuções (definição 3.1.6). A árvore de computação é formada simplesmente percorrendo o autômato global (ou estrutura Kripke) que representa o sistema multi-agentes, a partir de seu estado inicial. Uma árvore de computação descreve todas as execuções possíveis no conjunto de execuções R.

Sintaxe e Semântica para KCTL

Seguindo a forma usual adotada em vários trabalho correlatos (conforme descrito na seção 1.2), adotamos uma linguagem proposicional multi-modal, com uma modalidade de conhecimento K_i para cada agente i acrescida a linguagem de CTL.

Seja $j \in I\!N$ e $\{\mathcal{P}_k\}_{1 \leq k \leq j}$ o conjunto de conjuntos disjuntos de proposições. A linguagem de fórmulas KCTL é definida como:

$ForKCTL(j, \{\mathcal{P}_k\}_{1 \leq k \leq j})$ é o menor conjunto For de fórmulas tal que:

- $p \in For$ se e somente se existe k tal que $1 \leq k \leq j$ e $p \in \mathcal{P}_k$,
- $K_i(\phi) \in For$ se e somente se $1 \leq i \leq j$ e $\phi \in For$,
- $\{\neg\phi_1, \phi_1 \vee \phi_2, \exists \mathsf{X}\phi_1, \exists \mathsf{G}\phi, \exists[\phi_1 \mathsf{U} \phi_2]\} \subseteq For$ se e somente se $\{\phi_1, \phi_2\} \subseteq For$.

Considerando a definição de "caminho" em um modelo global de Kripke a semântica de fórmulas p, $\{\neg\phi_1,\ \phi_1 \vee \phi_2,\ \exists \mathsf{X}\phi_1,\ \exists \mathsf{G}\phi,\ \exists[\phi_1 \mathsf{U} \phi_2]\}$ de KCTL é definida como na semântica de CTL.

A semântica das modalidades de conhecimento K_i é dada da forma usual, baseada nas relações epistêmicas de acessibilidade $\sim_i$ – conforme apresentado na seção 2.1. Dois estados s e t de S estão relacionados através de $\sim_k$ se e somente se o agente i não é capaz de distinguir entre eles. Dizemos que um agente "sabe" um fato ϕ em um estado s se e somente se ϕ vale em todos os estados (ou mundos possíveis, seguindo a terminologia usual) que o agente considera possível no estado s.

Dado um estado s de um modelo Kripke $\mathfrak{M} = \langle S, S_0, R, \{\sim_k\}_{1 \leq k \leq j}, \cup_{k=1}^{j} \mathcal{P}_k, \mathcal{L} \rangle$:

$\mathfrak{M}, s \models K_i(\phi)$ se e somente se para todo $s' \in S$ tal que $s' \sim_i s$, $\mathfrak{M}, s' \models \phi$

4.1.3 Processo de verificação para propriedades epistêmicas

Agora apresentamos algoritmos para o problema de checagem de fórmulas KCTL em modelos globais de Kripke.

O processo é o mesmo usualmente empregado para verificação explícita de modelos, apresentado na seção 3.1.3: "Para verificar se uma fórmula de KCTL f é satisfeita em um dado estado s de S, o processo consiste em rotular cada estado $s \in S$ com o conjunto *label(s)* de subfórmulas de f que são verdadeiras em s. Inicialmente, *label(s)* é $\mathcal{L}(s)$. A seguir, o algoritmo passa por uma série de iterações, adicionando subfórmulas a *label(s)*. Durante a i-ésima iteração, subfórmulas com $i - 1$ operadores KCTL aninhados são processados e adicionados ao rótulo dos respectivos estados onde são satisfeitos. Ao final, $\mathfrak{M}, s \models f$ se e somente se $f \in$ *label(s)*".

Para os estados intermediários do algoritmo, é necessário tratar sete casos: fórmulas atômicas, $\neg, \vee, \exists\mathsf{X}, \exists\mathsf{G}, \exists\mathsf{U}$ e K. Os seis primeiros casos são os mesmos de CTL, descritos na seção 3.1.3, onde são discutidos também os algoritmos dados em [PCG00] para tratamento destes casos com complexidade $O(|S| + |R|)$.

Daremos especial tratamento ao sétimo caso, que define como o operador de conhecimento será tratado.

Seguindo o significado intuitivo e a semântica definida para os operadores K_i em KCTL, para verificar uma fórmula da forma $K_i f$ é necessário observar os estados indistinguíveis para o agente i no estado em questão, ligados pela relação de equivalência $\sim_i$.

O processo consiste em: primeiro, encontrar o conjunto de todos os estados s rotulados com f. Então, para cada estado s encontrado, checar recursivamente se todos os estados t relacionados com s (o estado corrente) através de $\sim_i$ (todo t tal que $s \sim_i t$) está rotulado com f. Se este é o caso, então rotula-se todos os estados ligados pela relação (o estado corrente s e todo t tal que $s \sim_i t$) com $K_i f$.

Apesar de recursivo, este processo é linear no número de estados de S. Este é o caso porque $\sim_i$ é uma relação de equivalência, resultando que $s \sim_i t$ se e somente se $t \sim_i s$,

e que cada estado em S pertence a uma única classe de equivalência definida por $\sim_i$. Uma vez iniciado o processo de rotulação, um estado é elegido e todos os estados da mesma classe de equivalência a qual pertence este estado são examinados e rotulados ou com $K_i f$ ou com $\neg K_i f$. Cada estado é examinado uma única vez. Isto significa que a complexidade de tratar o operador K_i é da ordem de $O(|S|)$.

Uma vez que estão descritos os algoritmos para tratar todos os sete casos, para tratar uma fórmula arbitrária de f de KCTL basta aplicar sucessivamente o processo de rotulação de estados para as subfórmulas de f, iniciando com a menor e mais aninhada, e trabalhando de dentro para fora até que f esteja inteiramente verificada. O processo completo leva tempo $O(|f| \cdot (|S| + |R|)$, ou seja, o tratamento do operador modal de conhecimento não aumenta a complexidade do problema de verificação para CTL.

O algoritmo em pseudo-código para tratamento do operador de conhecimento em fórmulas da forma $K_i f$ é descrito a seguir.

Function `CheckK` (*G, f, i*)

G [in parameter] é um autômato global;
f [in parameter] é a fórmula a ser avaliada;
i [in parameter] é o identificador do agente;
L é um conjunto de estados, inicialmente vazio;
begin
  $L := \emptyset$; $S := \{s | f \in \mathit{label}(s)\}$
  foreach *state* s *in* S **do**
    if $K_i f \notin$ label(s) *and* $\neg K_i f \notin$ label(s) **then**
      if *RecursiveCheckK(G, f, i, L, s)* **then**
        foreach *state* t *in* L **do**
          $label(t) := label(t) \cup \{K_i f\}$
        end
        else
          $label(t) := label(t) \cup \{\neg K_i f\}$
        end
      end
    end
  end
end

Function `RecursiveCheckK(`*G, f, i, L, s*`)`

```
G [in parameter] é um autômato global;
f [in parameter] é a fórmula a ser avaliada;
i [in parameter] é o identificador do agente;
L é um conjunto de estados, inicialmente vazio;
s [in parameter] é um estado do modelo;
begin
    if s ∉ L then
        if f ∈ label(s) then
            L := L ∪ {s}
            foreach state t ∈ G, so that t ∼_i s do
                if RecursiveCheckK(G, f, i, L, t) then
                    return True
                    else
                        return False
                    end
                end
            end
        end
        else
            return True
        end
end
```

4.1.4 Exemplo: Protocolo do Bit Alternado

O *protocolo do bit alternado* é um famoso protocolo de comunicação básico, comumente usado como caso de teste para formalismos algébricos ou para análise de sistemas concorrentes. Ele consiste em um sistema distribuído com três partes (agentes) componentes: um agente transmissor, chamado **sender**, um agente receptor, chamado **receiver** e um canal de comunicação – **communication channel**. O transmissor possui um conjunto de mensagens para enviar ao receptor através do canal de comunicação. Porém, o canal não é confiável, podendo perder mensagens que passem por ele antes que sejam entregues ao destinatário.

Assume-se que o canal pode transportar apenas uma mensagem de cada vez e que é bidirecional, isto é, pode transportar mensagens do transmissor para o receptor e vice versa, do agente receptor para o transmissor.

O protocolo inicia com o agente transmissor selecionando a primeira mensagem a enviar. Esta mensagem é estendida com um bit de controle (inicialmente 0) para compor um pacote de envio e este pacote é logo enviado através do canal de comunicação. Uma vez que o envio do pacote começa, o agente transmissor também inicia um temporizador. Quando o temporizador chega a zero, o transmissor assume que o pacote foi perdido (*timeout*) e o retransmite.

Ao receber o pacote, o canal de comunicação pode se comportar de duas formas: transmitir corretamente o pacote, ou perder o pacote durante a transmissão.

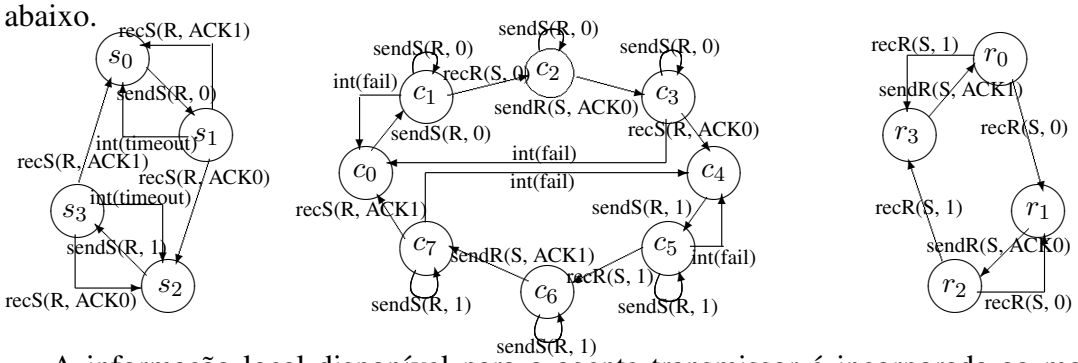
Se o pacote não for perdido, o receptor recebe o pacote do canal. O receptor checa então o bit de controle no pacote – se o bit corresponde ao bit de controle esperado (bit interno do receptor) então o recebimento da mensagem é confirmado, isto é, o receptor manda uma mensagem de confirmação (*acknowledgement*) contendo o mesmo bit de controle da mensagem recebida ao transmissor, através do mesmo canal de comunicação. O receptor inverte o valor do bit de controle (no caso inicial, de zero para um) e espera pelo próximo pacote. Se o bit do pacote recebido estiver diferente do esperado, o receptor envia uma confirmação negativa (pacote com o bit invertido), e aguarda a retransmissão do pacote.

Como o canal não é confiável, há ainda a chance de que a mensagem de confirmação seja perdida. Caso isto aconteça, não há nada a fazer a não ser esperar que o temporizador chegue a zero e que o transmissor então retransmita o pacote, acreditando que o anterior não foi recebido. Isto acarretará que uma nova confirmação seja enviada, igual a que foi perdida, e o receptor ignore o conteúdo (já anteriormente recebido) da mensagem.

O processo assim continua até que o agente transmissor receba a confirmação de uma transmissão bem sucedida, o que ocorre quando o bit de confirmação recebido é o mesmo que o bit de controle interno do agente transmissor. Se o bit não confere, a mensagem de confirmação é ignorada. Após uma transmissão bem sucedida, o agente transmissor inverte o valor do seu bit de controle, seleciona a próxima mensagem a ser enviada e o processo recomeça.

Especificação do sistema

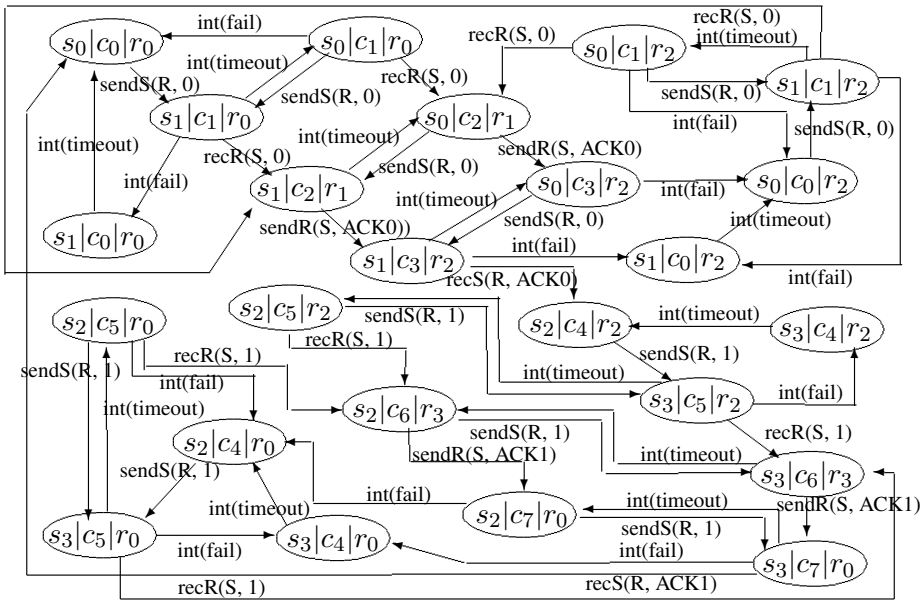
Cada agente é modelado como um autômato, que representa o comportamento do agente e a informação local (codificada no estado). Os autômatos para os agentes transmissor, receptor e canal de comunicação estão representados respectivamente nas figuras abaixo.



A informação local disponível para o agente transmissor é incorporada ao modelo como um conjunto de proposições que valem em cada estado (o mesmo para o agente receptor). Considera-se que o canal de comunicação não guarda nenhuma informação além do seu estado local (mensagem em transmissão). Segue uma lista das informações válidas para os agentes transmissor e receptor em cada estado.

Sender	**Receiver**
`S0: sent_msg_bit_1` `received_Ack1`	`R0: receiving_msg_bit_0` `sent_Ack1`
`S1: sending_msg_bit_0` `receiving_Ack0`	`R1: received_msg_bit_0` `sending_Ack0`
`S2: sent_msg_bit_0` `received_Ack0`	`R2: receiving_msg_bit_1` `sent_Ack0`
`S3: sending_msg_bit_1` `receiving_Ack1`	`R3: received_msg_bit_1` `sending_Ack1`

O autômato global correspondente ao sistema é obtido a partir da composição dos autômatos locais, conforme a definição 4.1.5.



Verificando o modelo

Há muitas fórmulas interessantes sobre este modelo que podemos escrever em KCTL.

Esboçamos a seguir alguns resultados parciais do processo de verificação para uma fórmula KCTL envolvendo operadores temporais e de conhecimento: $G, (s_0|c_0|r_0) \models K_S K_R \neg \exists \mathsf{F} \neg (sending\_Ack0 \rightarrow \exists \mathsf{F} receiving\_bit\_1)$: No início, o agente transmissor sabe que o agente receptor sabe que sempre é o caso que se confirmar o recebimento do bit 0 eventualmente irá receber o bit 1?

Resultados para $sending\_Ack0$: Todos os estados globais que contém estado local r_1.

Resultados para $receiving\_msg\_bit\_1$: Todos os estados globais que contém o estado local r_2.

Resultados para $\exists \mathsf{F} receiving\_msg\_bit\_1$: Todos os estados do modelo.

Resultados para $sending\_Ack0 \rightarrow \exists \mathsf{F} receiving\_bit\_1$: Todos os estados do modelo.

Resultados para $\neg sending\_Ack0 \rightarrow \exists \mathsf{F} receiving\_bit\_1$: Nenhum estado do modelo.

Resultados para $\exists \mathsf{F} \neg sending\_Ack0 \rightarrow \exists \mathsf{F} receiving\_bit\_1$: Nenhum estado do modelo.

Resultados para $\neg \exists \mathsf{F} \neg sending\_Ack0 \rightarrow \exists \mathsf{F} receiving\_bit\_1$: Todos os estados do modelo.

Assim sendo, a fórmula epistêmica $K_S K_R (sending\_Ack0 \rightarrow \exists \mathsf{F} receiving\_bit\_1)$ também vale em todos os estados do modelo, incluindo o estado inicial. Então o processo

de verificação retorna **true**.

A abordagem apresentada para construir um modelo global para sistemas multi-agentes a partir de especificações locais baseadas em autômatos possui a vantagem de que as relações de acessibilidade podem ser extraídas naturalmente (e geradas automaticamente) a partir do processo de construção do modelo. Os algoritmos para verificação de fórmulas envolvendo o operador modal de conhecimento também são feitos com base nas propriedades que esta construção acarreta, e não aumentam a complexidade do processo de verificação de uma fórmula CTL.

4.2 Verificação Formal de propriedades epistêmicas em Sistemas Probabilísticos Discretos

Após discutir os processos de modelagem e verificação de propriedades epistêmicas em sistemas não probabilísticos, voltamos nossa atenção para classes de sistemas que podem apresentar eventos probabilísticos. Investigamos como estender os resultados anteriores para modelos probabilsíticos de tempo discreto. Levamos em conta que o comportamento interno dos agentes do sistema pode ser completamente descrito por um modelo probabilístico, porém o agente pode fazer uso de ações não probabilísticas para interagir com outros agentes. Consideramos que esta é uma opção razoável pois não requer nenhuma co-dependência (ou normalização de probabilidades) entre os modelos para dois agentes distintos do sistema.

Conforme apresentado na seção 3.2, o modelo MDP pode ser visto como uma composição de modelos DTMC. Cada DTMC possúi todas as medidas probabilísticas bem definidas, representando o comportamento probabilístico de um agente.

Discutimos como incorporar conhecimento na linguagem padrão para raciocinar sobre esta classe de modelos, PCTL (descrita na seção 3.2.2). Em seguida, propomos um processo de verificação automática de especificações envolvendo conhecimento.

4.2.1 Modelagem de Sistemas multi-agentes probabilísticos de tempo discreto

Cada agente tem seu comportamento (probabilístico) modelado por uma cadeia de Markov de tempo discreto (*Discrete-Time Markov Chain* – DTMC), um autômato onde as transições possuem probabilidades associadas.

Uma DTMC é uma família de variáveis randômicas $\{X(k)|k = 0, 1, 2, \cdots\}$ onde as observações são feitas em passos de tempo discretos. Os valores assumidos por $X(k)$ são chamados *estados*. O conjunto de todos os estados, a que chamamos *espaço de estados*, é discreto. Para o escopo deste trabalho, restringimos nossa atenção para o caso onde o espaço de estados é finito. Uma DTMC obedece a propriedade de Markov, de forma que a probabilidade de o sistema estar em um determinado estado s no tempo k depende apenas no estado onde estava o sistema no tempo imediatamente anterior, $k-1$. Para todo inteiro

$k \geq 0$ e estados $s_0, \cdots, s_k$:

$$P[X(k) = s_k | X(k-1) = s_{k-1}, \cdots, X(0) = s_0] = P[X(k) = s_k | X(k-1) = s_{k-1}]$$

Além disso, consideraremos apenas DTMCs homogêneas, o que significa que a probabilidade de uma transição é independente do tempo em que a transição ocorre. A descrição formal do modelo DTMC foi feita na definição 3.2.1. Resulta que para descrever uma DTMC homogênea cujo espaço de estados é finito é suficiente dar uma matriz que diga para qualquer par de estados a probabilidade de uma transição entre eles. Uma vez que cada agente é descrito por um DTMC, um sistema multi-agentes pode ser modelado por uma composição paralela de DTMCs.

A literatura oferece duas abordagens principais para o tratamento de não determinismo na composição de DTMCs: "totalmente probabilístico" e "alternante". Abordagens totalmente probabilísticas substituem não-determinismo por distribuições de probabilidade. Nas abordagens alternantes existe uma diferenciação clara entre escolhas (puramente) não-determinísticas e probabilísticas, de forma que ambas são permitidas. Essa separação permite definir composição paralela assíncrona de modelos, uma vez que o não-determinismo pode ser usado para expressar liberdade de escolha através da intercalação de ações.

A noção de não-determinismo é essencial na modelagem de sistemas multi-agentes, da mesma forma que é essencial para para álgebra de processos em geral. Não-determinismo é útil para modelar conceitos importantes como por exemplo:

- Liberdade de implementação: um modelo abstrato para sistema multi-agentes deve permitir liberdade de implementação, isto é, para um estado onde duas transições podem ser escolhidas não-deterministicamente, uma determinada implementação pode oferecer apenas uma dentre as duas transições.

- Liberdade de escolha: Vários processos executam em paralelo e existe liberdade de escolha em relação a transição que será executada a seguir.

- Ambiente externo: Ações externas representam possíveis interações com um processo externo por meio de sincronização. A capacidade de interação deste ambiente influencia em como a escolha é determinada.

Dito isso, neste trabalho adotaremos modelos capazes de representar não-determinismo (puro) e probabilidade.

Adaptaremos o modelo para sistemas concorrentes de [LMWF94]. para o caso probabilístico, utilizando os mesmos conceitos para definir a composição de um conjunto de DTMCs. O modelo de [LMWF94] foi discutido na seção 4.1.1. Considerando que podem haver ações de sincronismo entre os agentes, temos que criar mecanismos para que estas ações sejam corretamente tratadas na composição. Relembrando que as ações de sincronização estão sobre a influência de mais de uma componente do modelo global, é importante fornecer mecanismos que permitam liberdade de escolha e de implementação para as situações de sincronismo. Optamos por evitar uma normalização da probabilidade entre os diferentes agentes envolvidos em ações de sincronismo, e a opção trivial é atribuir a elas probabilidade 1.

Aproveitamos o conceito de assinatura de ações (denifinção 4.1.1) e o adaptamos para modelos DTMC.

Definição 4.2.1 *(Assinatura de ações)*

Uma assinatura de ações $\mathfrak{S}$ é um trio composto por três conjuntos de ações distintos dois a dois, chamados $in(\mathfrak{S}), out(\mathfrak{S})$, e $int(\mathfrak{S})$. As ações nesses conjuntos são chamadas respectivamente de ações de entrada, ações de saída, e ações internas de $\mathfrak{S}$. É também definido $ext(\mathfrak{S}) = in(\mathfrak{S}) \cup out(\mathfrak{S})$ como as ações externas de $\mathfrak{S}$, $local(\mathfrak{S}) = int(\mathfrak{S}) \cup out(\mathfrak{S})$ como as ações localmente controladas de $\mathfrak{S}$, e $acts(\mathfrak{S}) = in(\mathfrak{S}) \cup out(\mathfrak{S}) \cup int(\mathfrak{S})$ como as ações de $\mathfrak{S}$.

Definição 4.2.2 *(cadeia de Markov de tempo discreto DTMC com assinatura)*

Uma DTMC com assinatura é uma tupla $\mathcal{D} = (\mathfrak{S}, S, \bar{s}, \mathbf{P})$ onde:

- *$\mathfrak{S}$ é uma assinatura de ações*
- *S é um conjunto finito de estados;*
- *$\bar{s} \in S$ é o estado inicial;*
- *$\mathbf{P} : S \times \mathfrak{S} \times S \rightarrow [0, 1]$ é uma matriz de probabilidades de transição, tal que:*

– *para toda ação* $a \in ext(\mathfrak{S})$, $\mathbf{P}(s, a, s') = 1$, *para qualquer* s, s' *(todas as ações externas tem probabilidade* 1*);*

– $\sum_{s' \in S} \mathbf{P}(s, a, s') = 1$ *para todos os estados* $s \in S$ *e ações* $a \in \mathfrak{S}$.

Consideramos a definição de assinatura de ações compatíveis dada anteriormente (definição 4.1.3), e também a definição anterior de composição de assinatura de ações (definição 4.1.4), podemos definir a composição de um conjunto de DTMCs.

A composição de DTMCs gera um processo de decisão de Markov – MDP (*Markov Decision Process*), um autômato envolvendo transições probabilísticas e não determinísticas. Formalizamos a seguir a composição de um conjunto de DTMCs com assinatura.

Definição 4.2.3 *(Composição paralela de DTMCs com assinatura: MDP global)*

Uma coleção $\{\mathfrak{D}_i = \langle \mathfrak{S}_i, S_i, (\bar{s_i}), \mathbf{P}_i \rangle\}_{i \in I}$ *de DTMCs com assinatura é dita compatível se suas assinaturas de ações são compatíveis, de acordo com a definição 4.1.3. A composição* $\mathfrak{M}_I = \Pi_{i \in I} \mathfrak{D}_i$ *de uma coleção compatível finita de DTMCs com assinatura* $\{\mathfrak{D}_i\}_{i \in I}$*, possui as seguintes componentes:*

- $S = \Pi_{i \in I} S_i$,
- $s_0 = \Pi_{i \in I} (\bar{s_i})$,
- $\mathfrak{S} = \Pi_{i \in I}\ \mathfrak{S}_i$, *onde* $\Pi_{i \in I}\ \mathfrak{S}_i$ *denota a composição, segundo a definição 4.1.4, de um conjunto compatível* $\mathfrak{S}_i, i \in I$ *de assinaturas de ações.*
- $R :\subseteq S \times acts(\mathfrak{S}) \times S \rightarrow [0, 1]$ *é uma função de transição probabilística, construída a partir das matrizes de probabilidades de transição das DTMCs compostas, de forma que para todo* $i \in I$*:*

  – *se* $\pi \in \mathfrak{S}_i$*, então* $R[(s', \pi, s)] = P_i(s'[i], \pi, s[i])$*, e*

  – *se* $\pi \notin \mathfrak{S}_i$*, então* $R[(s', \pi, s)] = 0$ *sempre que* $s'[i] \neq s[i]$,

  onde a notação $[i]$ *representa a* i*-ésima componente do vetor de estados* s.

É fácil observar que um MDP global corresponde a um MDP descrito na definição 3.2.3. Isto significa que podemos estender a definição 3.2.4 de MDP rotulado para MDP global rotulado.

Definição 4.2.4 *(MDP global rotulado)*

Seja AP um conjunto de proposições atômicas. Um MDP global rotulado é um trio ordenado $(\mathfrak{M}_I, \mathcal{L}, c)$ onde:

- *$\mathfrak{M}_I$ é um MDP global;*
- *$\mathcal{L} : 2^{AP}$ é a função que associa a cada estado um subconjunto de proposições;*
- *$\mathbf{c} : S \times \mathfrak{S} \rightarrow \mathbb{R}^{\geq 0}$ é a função de custo, que associa custos não negativos às transições.*

Além disso, as definições de caminho em um MDP (definição 3.2.5) e adversário em um MDP (3.2.6), se aplicam também para MDP global, e por conseqüência também as definições 3.2.7, 3.2.8, 3.2.9 e 3.2.10.

Adversários são definidos para tratar escolhas não determinísticas (lembrando que um "adversário" é uma função mapeando cada caminho finito de um MDP em uma distribuição probabilística disponível para o último estado do caminho). O comportamento de um MDP sobre o controle de um dado adversário é puramente probabilístico.

A composição paralela descrita acima reflete uma composição assíncrona, de forma que as distribuições probabilísticas referentes a cada uma das DTMCs envolvidas são incorporadas no MDP resultante de forma independente. A única intersecção possível entre distribuições advindas de diferentes DTMCs são as ações de sincronização, porém estas tem probabilidade 1 por definição.

As figuras 4.1 e 4.2 ilustram um MDP composto por duas DTMCs, respectivamente sem e com ações de sincronização.

Uma vez que os modelos locais para cada agente são compostos, o modelo resultante apresenta não determinismo, que advém do fato de as distribuições probabilísticas provenientes de cada agente permanecerem independentes (não serem normalizadas). Isso acontece porque no MDP não há informação indicativa de tempo, ou seja, de quanto tempo um agente pode levar para realizar uma transição ou ficar no mesmo estado. Um agente pode ser muito mais rápido do que o outro, ou ainda, o desempenho pode ser inconstante.

Para raciocinar sobre conhecimento nesta classe de sistemas probabilísticos, onde a velocidade relativa de execução entre os agentes é desconhecida, usaremos a mesma

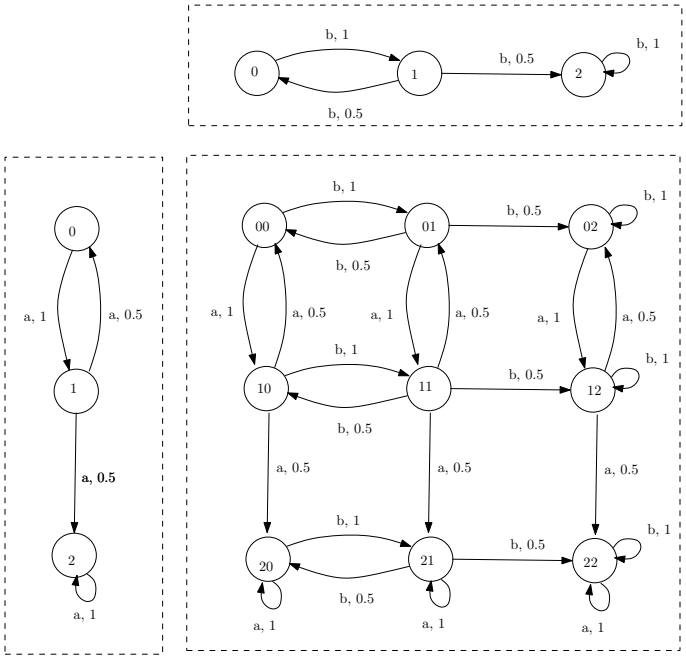


Figura 4.1: Um MDP formado pela composição de duas DTMCs.

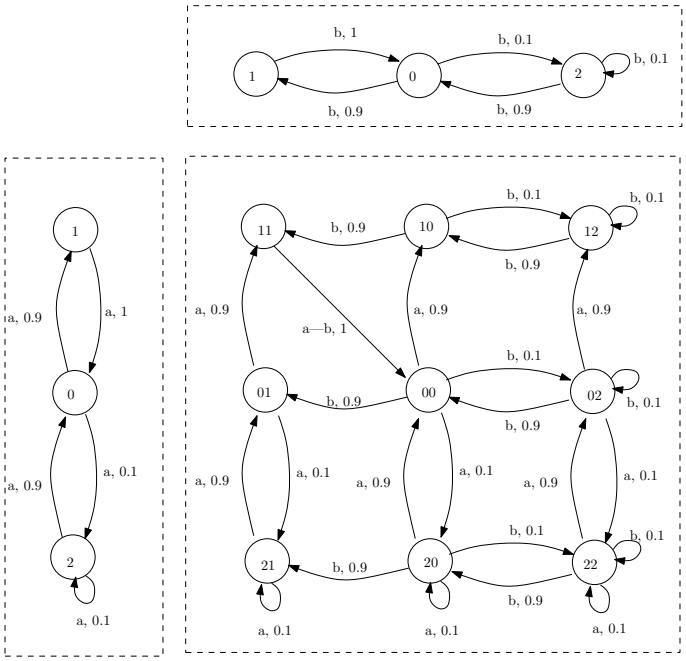


Figura 4.2: Um MDP formado pela composição de duas DTMCs, com uma ação sincronizada.

abordagem adotada anteriormente, para o modelo puramente não determinístico – as incertezas do modelo advém exatamente das opções não determinísticas, geradas quando ocorre a composição dos modelos correspondentes a cada agente do sistema, refletindo o desconhecimento de um agente acerca do comportamento dos outros agentes.

Intuitivamente, os conjuntos de estados indistinguíveis para um agente são definidos considerando que o estado local do agente é o mesmo, ou seja, quando o estado global muda e o estado local permanece o mesmo, o que é ocasionado pelo não determinismo.

Definição 4.2.5 *(Relação de acessabilidade $\sim_i$ para o agente i)*

Seja $\mathfrak{M}_I$ um MDP global rotulado. A relação de acessabilidade $\sim_i \in S \times S$ para cada agente i é a menor relação de equivalência contendo todos os pares (s', s), $s', s \in S$ tais que $s'[i] = s[i]$.

Definição 4.2.6 *(MDP $\sim$global)*

Um MDP $\sim$global $\mathfrak{M}_I^{\sim}$ é um MDP global rotulado acrescido de relações de acessabilidade conforme descritas na definição 4.2.5, para todo agente $i \in I$.

4.2.2 Especificação: lógica PCTL com operador de conhecimento K–PCTL

Conforme discutido na seção 3.2.2, para escrever especificações de MDPs, usa-se uma extensão probabilística da lógica CTL, chamada de lógica probabilística de tempo ramificado, *Probabilistic Computation Tree Logic* – PCTL.

Ao dar o enfoque de sistemas multi-agentes ao modelo e introduzir as relações $\sim_i$, podemos enriquecer a linguagem para nos permitir expressar também propriedades epistêmicas. Para raciocinar sobre conhecimento e probabilidade, acrescentaremos a linguagem de PCTL o operador modal de conhecimento K_i, para cada agente i. A linguagem resultante será chamada de K–PCTL.

Sintaxe e Semântica para K–PCTL

Adotamos uma linguagem proposicional multi-modal, com uma modalidade de conhecimento K_i para cada agente i acrescida a linguagem de PCTL.

Seja AP o conjunto de proposições atômicas, e I o conjunto de agentes. A linguagem de fórmulas KCTL é definida como:

$\phi ::= true \mid a \mid \neg\phi \mid \phi \wedge \phi \mid K_i\phi \mid P_{\bowtie p}[\psi] \mid \mathcal{E}_{\bowtie c}[\phi]$,

$\psi ::= \phi \mid \phi \mathcal{U}^{\leq k} \phi \mid \phi \mathcal{U} \phi$

onde a é uma proposição atômica de AP, $i \in I$, $\bowtie \in \{\leq, <, \geq, >\}$, $p \in [0,1]$, $c \in \mathbb{R}^{\geq 0}$ e $k \in \mathbb{N}$.

Fórmulas K–PCTL são avaliadas sobre MDPs $\sim$globais. A semântica de fórmulas $true$, a, $\neg\phi$, $\phi \wedge \phi$, $P_{\bowtie p}[\psi]$, $\mathcal{E}_{\bowtie c}[\phi]$ de K–PCTL, onde $\psi ::= \phi \mid \phi \mathcal{U}^{\leq k} \phi \mid \phi \mathcal{U} \phi$, é definida como na semântica de PCTL (seção 3.2.2).

A semântica das modalidades de conhecimento K_i é dada da forma usual, baseada nas relações epistêmicas de acessabilidade $\sim_i$ – conforme apresentado na seção 2.1.

Dado um estado s de um MDP $\sim$global $\mathfrak{M}_I^{\sim}$:

$$\mathfrak{M}_I^{\sim}, s \models K_i(\phi) \quad \text{se e somente se} \quad \begin{array}{l}\text{para todo } s' \in S \text{ tal que } s' \sim_i s, \\ \mathfrak{M}_I^{\sim}, s' \models \phi\end{array}$$

4.2.3 Processo para verificação de propriedades epistêmicas

Para verificar se uma fórmula f de PCTL é satisfeita em um estado s de um MDP $\sim$ composto, utiliza-se o processo recursivo descrito na seção 3.2.3, onde cada operador é tratado em separado.

Para verificar uma fórmula epistêmica, da forma $K_i(f)$, devemos olhar para os estados indistinguíveis para o agente i, relacionados através da relação de equivalência $\sim_i$. Como a relação $\sim_i$ foi definida da mesma forma que para o caso não probabilístico, o processo é o mesmo descrito na seção 4.1.3.

Considerando que a complexidade de processar o operador K_i é $O(|S|)$ e que a complexidade do processo de verificação de PCTL sobre modelos MDP é linear no tamanho da fórmula e polinomial no tamanho do modelo (seção 3.2.4), concluímos que a inclusão do operador de conhecimento em K–PCTL não acarreta aumento da complexidade do processo completo de verificação de modelos MDP.

4.2.4 Exemplo: Leitura combinada de imagens

Consideramos um sistema simples com três agentes. Dois leitores de imagens, e um transmissor que coleta as leituras feitas pelos agentes leitores. Os agentes leitores de imagem, que chamaremos de **agente**A e **agente**B, têm funcionamento idêntico e comportamento probabilístico. Cada um possui apenas dois estados, que denotam as duas situações possíveis que o agente pode estar: realizando uma leitura (estado 1), ou calibrando (estado 0). As ações possíveis são calibrar (**aCal** e **bCal**, respectivamente para o agente A e B) , ler (**aRead** e **bRead**), e sincronizar (**sincA** e **sincB**). As ações calibrar e ler são probabilísticas, enquanto a ação de sincronizar tem probabilidade 1.

O transmissor simplesmente coleta as leituras dos agentes A e B para futura transmissão, primeiro a do agente A, em seguida a do agente B. As ações de sincronização entre os agentes tem probabilidade 1, conforme exigido pelo modelo definido. A transmissão não será representada no modelo, por simplicidade. Os modelos locais de cada agente bem como o modelo correspondente ao sistema G_{ABC} composto pelos agentes A, B e C é mostrado na figura 4.3.

Consideramos que nos estados 0 dos agentes A e B, respectivamente as proposições $calA$ e $calB$ são válidas, indicando que os agentes estão calibrando.

A fórmula K–PCTL $K_C P_{\leq 0,1} calA \wedge calB$ significa que o agente C sabe que a probabilidade de que ambos os agentes A e B estejam calibrando é menor que $0,1$. Esboçamos a seguir resultados parciais do processo de verificação desta fórmula envolvendo operadores de probabilidade e de conhecimento no estado $[000]$ do modelo global G_{ABC} descrito anteriormente:

$G_{ABC}, [000] \models K_C P_{\leq 0,1} calA \wedge calB$ se e somente se em todos os estados indistinguíveis de $[000]$ do ponto de vista do agente C, $P_{\leq 0,1} calA \wedge calB$ é válido. Recordemos que o estados indistinguíveis neste caso são aqueles onde o agente C possui o mesmo estado local que em $[000]$, ou seja, o estado local de C é 0. São estes os estados $[000]$, $[010]$, $[100]$ e $[110]$ de G_{ABC}. Verificamos se para cada um destes estados é caso que $P_{\leq 0,1} calA \wedge calB$ vale.

Seguindo a função de valoração definida para as fórmulas probabilísticas, é possível verificar que $P_{\leq 0,1} calA \wedge calB$ vale em todos os estados $[000]$, $[010]$, $[100]$ e $[110]$. Portanto, é o caso que $G_{ABC}, [000] \models K_C P_{\leq 0,1} calA \wedge calB$.

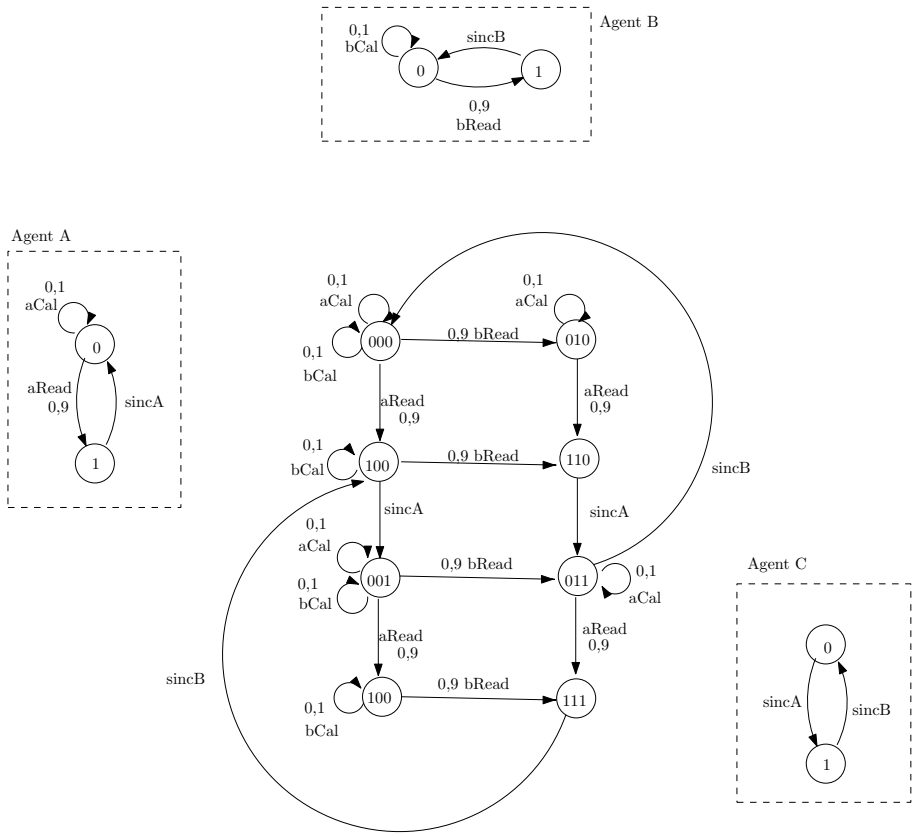


Figura 4.3: MDP G_{ABC} formado pela composição dos agentes A, B e C. Cada estado é denominado por um terno ordenado onde a primeira componente representa o estado local do agente A, a segunda componente representa o estado local do agente B e a terceira componente representa o estado local do agente C.

Consideramos agora o conhecimento do agente A sobre este fato, ou seja, o conhecimento de A sobre conhecimento do agente C de que $P_{\leq 0,1} calA \wedge calB$, no estado $[000]$: queremos saber se é o caso que $G_{ABC}, [000] \models K_A K_C P_{\leq 0,1} calA \wedge calB$. Os estados indistinguíveis do estado $[000]$ para o agente A são: $[000]$, $[010]$, $[001]$, $[011]$. Para cada um, analisamos os estados indistinguíveis para o agente C:

- $[000]$: são indistinguíveis sob o ponto de vista do agente C: $[000]$, $[010]$, $[100]$ e $[110]$;
- $[010]$: são indistinguíveis sob o ponto de vista do agente C: $[000]$, $[010]$, $[100]$ e $[110]$;

- $[001]$: são indistinguíveis sob o ponto de vista do agente C: $[001]$, $[011]$, $[101]$ e $[111]$;

- $[011]$: são indistinguíveis sob o ponto de vista do agente C: $[001]$, $[011]$, $[101]$ e $[111]$;

Ou seja, para todos os estados do modelo é necessário verificar se é o caso que $P_{\leq 0,1} calA \wedge calB$ vale. Acontece que para os estados que antecedem uma ação de sincronismo, que acontece com probabilidade 1, $P_{\leq 0,1} calA \wedge calB$ não vale, uma vez que sempre após o sincronismo uma calibração acontece. Dessa forma, não é o caso que $G_{ABC}, [000] \models K_A K_C P_{\leq 0,1} calA \wedge calB$, ainda que $G_{ABC}, [000] \models K_C P_{\leq 0,1} calA \wedge calB$. A explicação intuitiva é que o agente C realiza ação de sincronismo com o agente B, e o agente A não tem idéia do momento em que esta ação é realizada, de forma que considera o estado $[011]$, que antecede a execução de uma ação $sincB$, indistinguível de $[000]$. Isto não ocorre para o agente C, que tem controle local sobre a execução de $sincB$.

4.3 Verificação Formal de propriedades epistêmicas em Sistemas Probabilísticos de tempo contínuo

Abordaremos a seguir modelos probabilísticos de tempo contínuo para sistemas distribuídos, baseados em Cadeias de Markov de Tempo Contínuo (CTMC). O modelo de CMTC foi discutido na seção 3.3.1. Nesta classe de modelos, as distribuições de probabilidade são contínuas, mais especificamente exponenciais – transições Markovianas descrevem retardos exponencialmente distribuídos para se passar aos estados sucessores.

Primeiramente apresentaremos um processo para obter especificações globais a partir de composição de submodelos estocásticos de tempo contínuo. Em seguida, discutiremos como incorporar conhecimento ao modelo, levando em conta as características que o modelo construído apresenta, considerando em especial o tratamento adequado do não determinismo que pode ser introduzido pela composição do sistema a partir de especificações para cada agente. A seguir, enriqueceremos a linguagem padrão para raciocinar sobre propriedades temporais e probabilísticas em modelos probabilísticos de tempo contínuo– Lógica estocástica de tempo contínuo CSL (apresentada na seção 3.3.2), com operadores epistêmicos. Ao final, propomos um processo de verificação de modelos para a linguagem enriquecida.

4.3.1 Modelagem de Sistemas multi-agentes probabilísticos de tempo contínuo

Cadeias de Markov de tempo contínuo (CTMCs) são um modelo clássico em análise de desempenho. Conforme visto na seção 3.3.1, as CTMCs modelam tempo contínuo e escolhas probabilísticas, através da especificação de taxas de transição de um estado para outro. As CTMCs não modelam não determinismo.

Abordagens composicionais para construção de modelos baseados CTMC são discutidas em diversos trabalhos ([GHR93], [BHR84], [HR99], [Hil96], [BG98]), porém a maioria substitui o eventual não determinismo acarretado pela composição de processos interativos por distribuições de probabilidade. [RKNP04] menciona que uma CTMC pode simular processos concorrentes através das condições de competição que surgem de transições a partir de um mesmo estado com taxas diferentes. Essa é uma possível abor-

dagem, porém em si só esta opção não contempla modelos onde ações de sincronismo sejam possíveis. Dados os motivos descritos na seção 4.2.1, estamos interessados em modelos que contemplem não-determinismo puro e probabilidade.

Como estamos interessados em modelar sistemas multi-agentes em que a interação entre os agentes é um fator de grande importância, optaremos por um formalismo onde seja explicitamente oferecido um mecanismo de sincronização. Introduziremos a seguir o formalismo de Cadeias de Markov Interativas apresentado em [Her02]. O conceito de não determinismo está presente neste modelo, uma vez que não determinismo é um recurso valioso em métodos composicionais de especificação de modelos.

O modelo de Cadeias de Markov Interativas (Interactive Markov Chains – IMC) desenvolvido em [Her02] trata interação e consumo de tempo estritamente em separado, considerando que cada interação acontece instantaneamente uma vez que os agentes envolvidos na interação estejam prontos para realizá-la.

Definição 4.3.1 *(Cadeia de Markov Interativa – IMC)*
Uma IMC é uma quíntupla $(S, Act, \rightarrowtail, \twoheadrightarrow, P)$ *onde:*

- S *é um conjunto não vazio de estados,*
- Act *é um conjunto de ações,*
- $\rightarrowtail \subset S \times Act \times S$ *é um conjunto de transições rotuladas, também chamadas de transições interativas, e*
- $\twoheadrightarrow \subset S \times \mathbb{R}^+ \times S$ *é um conjunto de transições Markovianas,*
- $P \in S$ *é o estado inicial.*

As transições interativas podem ser de dois tipos: internas ou externas. Transições interativas internas, conforme veremos a seguir, são rotuladas por uma ação especial τ, chamada de ação interna. Estas transições estão completamente sobre o controle da IMC.

As transições interativas, a princípio, são como portas abertas para a interação com outros agentes, o que significa que o momento em que serão executadas não está sobre o controle da IMC. Consideramos que existe um tipo especial de ação, τ, que chamaremos

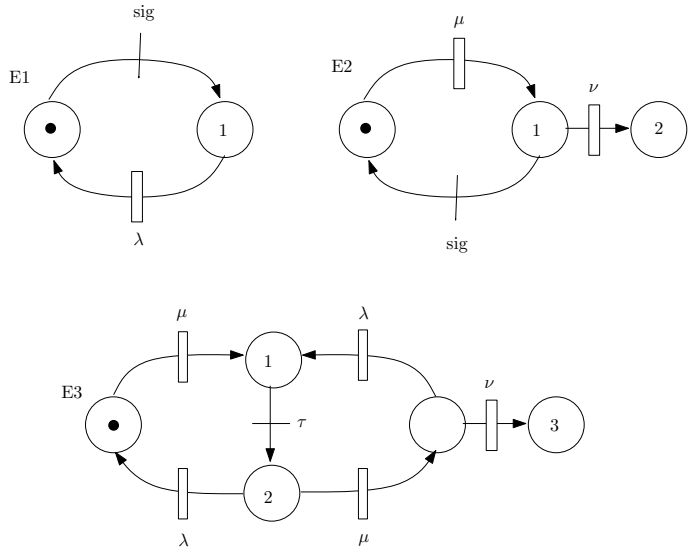


Figura 4.4: Três Cadeias de Markov Interativas.

de ação interna. As transições interativas rotuladas por τ representam transições interativas que estão sobre o controle da IMC, ou seja, transições interativas internas.

A figura 4.4 mostra três exemplos de IMCs. A primeira, $E1$, interage repetidamente no sinal sig e descansa durante um tempo exponencialmente distribuído após cada interação. $E2$ primeiro descansa por um intervalo de tempo exponencialmente distribuído (com taxa μ), depois interage no sinal sig e retorna ao estado inicial. Porém se esse sinal não ocorre em um intervalo de tempo dado (exponencialmente distribuído com taxa ν), $E2$ decide terminar sua execução. $E3$ apresenta uma transição interna τ, além de outras transições exponencialmente distribuídas.

Uma IMC que não possua transições interativas é equivalente a uma CTMC (definição 3.3.1). Da mesma forma, uma IMC sem transições Markovianas é equivalente a um Modelo Global de Kripke (definição 4.1.5).

Adotaremos nesta seção a mesma notação usada em [Her02], onde o estado inicial da IMC é utilizado para denominar a IMC em si.

Definiremos a seguir o operador de composição paralela (conforme [Her02]), que permitirá obter especificações compostas por duas ou mais IMCs. A composição deve se encarregar também de dar tratamento adequado as ações de sincronização, que são parâmetros da operação de composição paralela.

Definição 4.3.2 *(Composição paralela de IMCs)*

Sejam P e Q duas IMCs com espaço de estados S_P e S_Q, respectivamente. A composição paralela de P e Q com sincronização nas ações $a_1 \cdots a_n$ é uma IMC $(S, Act, \rightarrowtail, \twoheadrightarrow, P_{[a_1 \ldots a_n]}Q)$, onde:

- $S = \{P'_{[a_1 \ldots a_n]}Q' | P' \in S_P$ *e* $Q' \in S_Q\}$,
- Act *é a união dos conjuntos de ações de* P *e de que* Q,
- $\rightarrowtail$ *é a menor relação satisfazendo as seguintes regras para todo(s)* $P', P'' \in S_P$ *e todo(s)* $Q', Q'' \in S_Q$*:*

  Se $a \notin \{a_1 \cdots a_n\}$*:*

  - $P' \rightarrowtail^a P''$ *implica em* $P'_{[a_1 \ldots a_n]}Q' \rightarrowtail^a P''_{[a_1 \ldots a_n]}Q'$,
  - $Q' \rightarrowtail^a Q''$ *implica em* $P'_{[a_1 \ldots a_n]}Q' \rightarrowtail^a P'_{[a_1 \ldots a_n]}Q''$,

  Se $a \in \{a_1 \cdots a_n\}$*, então:*

  - $P' \rightarrowtail^a P''$ *e* $Q' \rightarrowtail^a Q''$ *implica em* $P'_{[a_1 \ldots a_n]}Q' \rightarrowtail^a P''_{[a_1 \ldots a_n]}Q''$
- $\twoheadrightarrow$ *é a menor relação satisfazendo as seguintes regras para todo(s)* $P', P'' \in S_P$ *e todo(s)*
  $Q', Q'' \in S_Q$*:*

  - $P' \twoheadrightarrow^\lambda P''$ *implica em* $P'_{[a_1 \ldots a_n]}Q' \twoheadrightarrow^\lambda P''_{[a_1 \ldots a_n]}Q'$
  - $Q' \twoheadrightarrow^\lambda Q''$ *implica em* $P'_{[a_1 \ldots a_n]}Q' \twoheadrightarrow^\lambda P'_{[a_1 \ldots a_n]}Q''$

De acordo com esta definição, transições Markovianas são intercaladas sem ajustar as respectivas taxas. A propriedade da ausênica de memória (discutida seção 3.3.1) garante que tal ajuste é desnecessário [Her02], uma vez que as distribuições são independentes – uma vez intercaladas, as distribuições "concorrerão" entre si para decidir qual delas será disparada primeiro, e a propriedade da ausência de memória garante que, para ambas as duas, o tempo transcorrido até o momento não é levado em conta. Uma vez que a primeira dispare, a cadeia muda para um estado que deveria representar o tempo restante para que

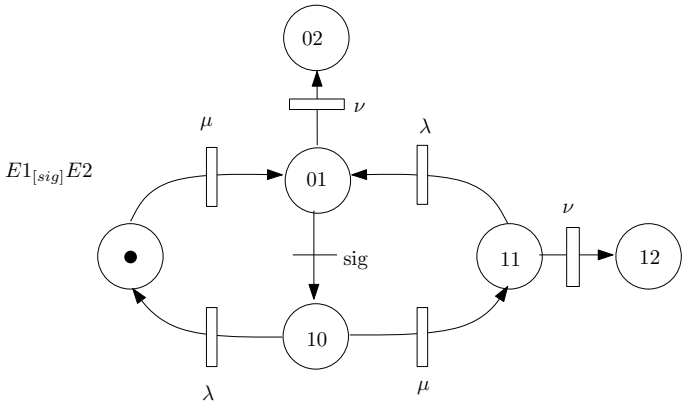


Figura 4.5: Composição paralela de duas IMCs.

a segunda cadeia dispare. Novamente, a propriedade da ausência de memória garante que este tempo é irrelevante para a distribuição, ou seja, não é necessário ajustar a taxa de transição.

Transições interativas são intercaladas não deterministicamente, caso nenhuma interação seja especificada. Caso contrário uma mudança simultânea de estados ocorre.

Cada estado de uma IMC composta $P_{[a_1 \ldots a_n]}Q$ pode ser considerado como um par ordenado de estados, formado pelos respectivos estados locais das IMCs P e Q envolvidas na composição. Sendo s um estado de uma IMC composta, e considerando a ordem de composição, usaremos a notação $s[i]$, onde $i \geq 1$ para nos referir ao estado local da IMC componente de ordem i utilizada na constituição do estado s por composição. O resultado da composição de duas IMCs é também uma IMC [Her02], de forma esta pode ser novamente composta com outra IMC qualquer R através de comoposição paralela: $(P_{[a_1 \ldots a_n]}Q)_{[b_1 \ldots b_n]}R$. Para evitar uma sobrecarga de notação, estendemos a notação $s[i]$ para os casos de composições sucessivas da forma mais simples possível: levando em conta a ordem em que as IMCs "atômicas" aparecem na composição, da esquerda para a direita.

A figura 4.5 ilustra a composição com sincronização na ação sig das IMCs $E1$ e $E2$ apresentadas na figura 4.4.

As ações interativas de uma IMC são como portas abertas a interação, ou seja, estão a mercê de interações com o ambiente. Para "encapsular" uma IMC de forma a bloquear interações que não forem explicitamente especificadas, [Her02] define outra operação

chamada abstração, cujos parâmetros são as ações que se deseja encapsular. As ações encapsuladas estão totalmente sobre o controle da IMC, ou seja, fechadas para interação externa.

Definição 4.3.3 *(Abstração de IMC)*

Seja P uma IMC com espaço de estados S_P. A abstração das ações $a_1 \cdots a_n$ em P é uma IMC $(S, Act, \rightarrowtail, \twoheadrightarrow, P_{(a_1 \cdots a_n)})$, onde

- $S = \{P'_{(a_1 \cdots a_n)} | P' \in S_P\}$,
- $\rightarrowtail$ *é a menor relação satisfazendo as regras seguintes para todo(s)* $P', P'' \in S_P$*:*

  Se $a \notin \{a_1 \cdots a_n\}$*, então:*

  - $P' \rightarrowtail^a P''$ *implica em* $P'_{(a_1 \ldots a_n)} \rightarrowtail^a P''_{(a_1 \ldots a_n)}$.

  Se $a \in \{a_1 \cdots a_n\}$*, então:*

  - $P' \rightarrowtail^a P''$ *implica em* $P'_{(a_1 \ldots a_n)} \rightarrowtail^\tau P''_{(a_1 \ldots a_n)}$.
- $\twoheadrightarrow$ *é a menor relação satisfazendo a seguinte regra, para todo(s)* $P', P'' \in S_P$*:*

  $P' \twoheadrightarrow^\lambda P''$ *implica em* $P'_{(a_1 \cdots a_n)} \twoheadrightarrow^\lambda P''_{(a_1 \cdots a_n)}$

A operação de abstração não possui nenhum impacto sobre as transições Markovianas. O seguinte teorema é conseqüência direta das definições anteriores:

Teorema 4.3.1 *IMCs são fechadas sobre as operações de composição paralela e abstração [Her02].*

A figura 4.6 mostra a IMC obtida após o encapsulamento da transição sig na IMC composta $E1_{[sig]}E2$ através de uma operação de abstração.

A idéia por trás do formalismo de IMCs é oferecer um modelo para composição de cadeias de Markov onde ações interativas e Markovianas são tratadas em separado. O comportamento das transições Markovianas obedece às taxas de transição a elas associadas. Já o comportamento das transições interativas vai depender do(s) comportamento(s) de quem têm controle sobre elas, segundo um princípio chamado de "máximo progresso".

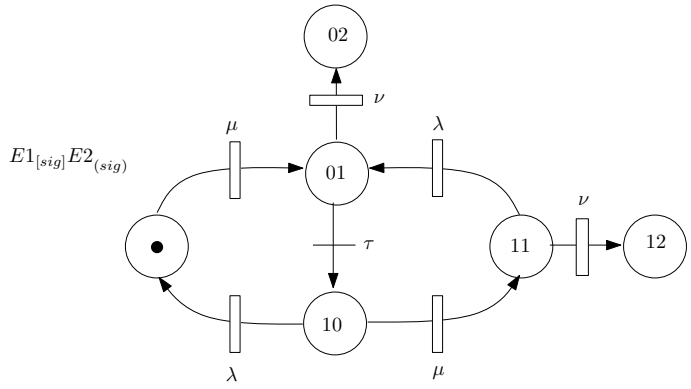


Figura 4.6: Abstração aplicada a uma IMC composta.

O princípio do máximo progresso dita que a ação é executada tão logo todos os que exercem controle sobre ela estão prontos para executá-la. Assim, transições internas (rotuladas por τ) são executadas imediatamente, sem permitir a passagem do tempo, pois estão sobre o total controle da IMC. Ações interativas externas podem ser retardadas indefinidamente aguardando a interação com o ambiente, e na ausência de informação adicional, o momento em que elas serão executadas é completamente imprevisível. A premissa do progresso maximal é amplamente utilizada em álgebra de processos de tempo real [NS92, Yi91, HR95].

Podemos usar IMCs para modelar composicionalmente sistemas multi-agentes. Para modelar um sistema multi-agentes usando IMCs, partimos de especificações IMC locais para cada agente, compomos as especificações locais utilizando a operação de composição paralela definida. Uma vez que o modelo global seja composto, todas as ações de sincronização devem ter sido respectivamente tratadas, e podem ser encapsuladas com a operação de abstração. Isso garante que o processamento do modelo global não depende de interações com o ambiente, e por conseguinte, que o modelo obtido contém toda a informação necessária para fazer uma análise ou verificação de seu comportamento.

Podemos então resumir as restrições que imporemos na modelagem de sistemas multi agentes:

- Em um modelo local todas as ações são Markovianas a menos de ações de sincronização;

- Um modelo global não contém ações interativas que não sejam ações encapsuladas (τ).

Definição 4.3.4 *(IMC global)*

Uma IMC $\Im_N$ é dita global caso seja obtida pela composição paralela de um conjunto N de IMCs – onde todas as ações interativas são de sincronização – e abstraída em todas as ações interativas.

Incorporando conhecimento

Podemos incorporar conhecimento ao modelo da mesma forma que foi feita nos dois casos anteriores, quando abordamos modelos não probabilísticos e probabilísticos com tempo discreto – os conjuntos de estados indistinguíveis para um agente são definidos considerando que o estado local do agente é o mesmo, ou seja, quando o estado global muda e o estado local permanece o mesmo. Neste modelo, esta situação pode ser ocasionada por transições Markovianas ou interativas.

Definição 4.3.5 *(Relação de acessabilidade $\sim_i$ para o agente i)*

Seja $\Im_N = (S, Act, \rightarrowtail, \twoheadrightarrow, P)$ uma IMC global. A relação de acessabilidade $\sim_i \in S \times S$ para cada agente $i \in N$ é a menor relação de equivalência contendo todos os pares (s', s), $s', s \in S$ tais que $s'[i] = s[i]$.

Definição 4.3.6 *(IMC $\sim$composta)*

Uma IMC $\sim$composta $\Im_{\sim N} = (S, Act, \rightarrowtail, \twoheadrightarrow, P, \sim_1, \sim_2, \cdots, \sim_n)$ é uma IMC global acrescida de relações de acessabilidade conforme descritas na definição 4.3.5, para cada agente $i \in N$.

Assim como nos modelos anteriores, o não determinismo (neste modelo representado pelas ações interativas) é um dos grandes reponsáveis pela introdução de incertezas em um sistema multi-agentes, incertezas sobre as quais é muito difícil associar qualquer estimativa, a menos que sejam fornecidas informações adicionais que possam influenciar a escolha não determinística em questão.

Falando especificamente de IMCs, a presença do não determinismo é uma conseqüência direta da separação entre retardos (estocásticos) e ações interativas. Levando em conta

a autonomia de processos e os mecanismos de sincronização, considerar estes dois conceitos em separado é razoável – não é realista impor uma taxa Markoviana a a uma ação interativa, mesmo porque essa taxa deveria levar em conta as o comportamento de todas as IMCs que influenciam na transição (e que podem não ser sabidos se não fazem parte de especificação). Porém em modelo de IMC global os impactos do não determinísmo no desempenho estocástico são minimizados, uma vez que uma IMC capaz de executar uma ação interna (encapsulada) é impedida de deixar que o tempo passe antes que essa ação seja executada, de acordo com a premissa do progresso maximal. Como conseqüência do progresso maximal, pode-se considerar que o comportamento das IMCs $E3$ (figura 4.4) e $E1_{[sig]}E2_{(sig)}$ (figura 4.6) é equivalente: como a transição τ é imediatamente executada tão logo o sistema entra no estado 1 de $E3$ na figura 4.4 e no equivalente 01 de $E1_{[sig]}E2_{(sig)}$ da figura 4.6, a transição ν a partir do estado 01 (figura 4.6) nunca terá a chance de ser executada. Isso acontece porque a execução de ν está vinculada a passagem do tempo, e a execução de τ é imediata.

O efeito acarretado pelo progresso maximal se aplica apenas a ações encapsuladas, uma vez que ações externas estão sujeitas a interação com o ambiente, e o momento em que a interação ocorrerá é governado pelo ambiente. Assim justificamos nossa opção por permitir apenas ações encapsuladas em uma IMC global. Uma IMC sem ações interativas externas (não abstraídas) têm grandes chances de ter seu comportamento temporal completamente regido por uma CTMC subjacente [Her02].

A possibilidade de minimizar a importância do não determinismo reflete uma redução das incertezas no modelo, o que pode influenciar a forma de modelar conhecimento, conforme veremos mais adiante. Nem sempre porém é possível eliminar completamente o não determinismo de um modelo IMC de forma que este se torne uma CTMC, uma vez que as cadeia de Markov de tempo contínuo são efetivamente modelos menos expressivos que as IMCs. Caso seja possível para uma IMC ter seu comportamento modelado por uma CTMC, tal CTMC pode ser obtida a partir do modelo IMC original através da aplicação de algoritmos para fatorar o espaço de estados, baseados no conceito de bissimilaridade fraca para IMCs proposto em [Her02].

Bissimilaridade fraca em modelos IMC

O procedimento de [Her02] baseia-se na possibilidade de encontrar formas de agregar o espaço de estados preservando o comportamento estocástico da IMC. Isto é atingido identificando estados pertencendo a uma classe de equivalência, e colapsando todos os estados da classe em um único. Logo, o procedimento trabalha agregando estados pertencentes a uma mesma classe de equivalência, e ajustando as transições envolvendo estes estados de acordo com as agregações feitas.

O objetivo do algoritmo é obter ao final uma CTMC capaz de modelar o comportamento estocástico da IMC original, para que se possa analisar o comportamento do sistema modelado utilizando os métodos tradicionais de análise de cadeias de Markov. As classes de equivalência que o algoritmo busca são classes de estados cujo comportamento estocástico é o mesmo. Um exemplo de estados que possuem o mesmo comportamento estocástico (apesar de não pertencerem a mesma IMC) são os estados 1 da IMC $E3$ da figura 4.4, e 01 da IMC $E1_{[sig]}E2_{(sig)}$ da figura 4.6, pelas razões que já foram discutidas anteriormente.

Levando em conta o progresso maximal, podemos negligenciar as transições internas que efetivamente não interfiram no comportamento estocástico da IMC. Para isso, a noção de passos observáveis de um processo é introduzida; um passo observável consiste uma única transição externa precedida e seguida por um número arbitrário (incluindo zero) de transições internas [Mil89]. Tecnicamente, isto é atingido derivando uma relação de transição "fraca" $\Longrightarrow$ a partir da relação de transição $\rightarrowtail$.

Definição 4.3.7 *(relação de transição fraca $\Longrightarrow$)*

Para ações internas τ, $\Longrightarrow^{\tau}$ é definida como o fecho transitivo $\rightarrowtail^{\tau *}$ da relação $\Longrightarrow^{\tau}$ para transições internas. Transições fracas externas são obtidas definindo $\Longrightarrow^{a}$ para denotar $\Longrightarrow^{\tau}\rightarrowtail^{a}\Longrightarrow^{\tau}$.

O tratamento de seqüências de transições internas precedendo uma transição Markoviana também é influenciado pela premissa de progresso maximal: transições Markovianas precedidas por uma seqüência de transições internas são relevantes apenas se emanam de um estado "estável", on seja, um estado de onde nenhuma outra transição interna é permitida.

Transições Markovianas são tratadas de forma a identificar as taxas acumuladas a atingir um estado ou uma classe de estados. Para isso, é definida uma função $\gamma_M : S \times 2^S \rightarrow \mathbb{R}^+$, que calcula a taxa acumulativa de atingir um conjunto de estados C a partir de um estado específico R: $\gamma_M(R, C) = \sum\{|\lambda|R \twoheadrightarrow^{\lambda} R' \text{ e } R' \in C|\}$, onde $\sum\{|\cdots|\}$ denota a soma de todos os elementos em um multi-conjunto (de taxas de transição), e $\{|\cdots|\}$ delimita tal conjunto.

A função γ_M é usada para acumular taxas de transições Markovianas que levam diretamente de um estado P a uma classe de equivalência específica C. Considerando o progresso maximal, é necessário acumular todas as taxas de transições Markovianas que levam a estados que podem, utilizando transições internas, evoluir para um estado em C. Para este propósito, é definido o fecho retroativo interno C^{τ} como o conjunto de estados que pode internamente evoluir para um elemento do conjunto C: $C^{\tau} = \{P'|\exists P \in C : P' \Longrightarrow^{\tau} P\}$.

A base formal do algoritmo é agrupar estados em classes de equivalência baseadas no conceito de bissimilaridade fraca em IMCs. Apresentamos a seguir a definição de bissimulação fraca em que se baseia o procedimento de [Her02] para fatorar o espaço de estados de uma IMCs.

Definição 4.3.8 *(Bissimulação fraca em IMCs)*

Uma relação de equivalência $\mathcal{R}$ sobre o conjunto de todas as IMCs é uma bissimlação fraca se e somente se $P\mathcal{R}Q$ implica para todo $a \in Act$:

1. *$P \Longrightarrow^{a} P'$ implica $Q \Longrightarrow^{a} Q'$ para algum Q' tal que $P'\mathcal{R}Q'$,*
2. *$P \Longrightarrow^{\tau} P'$ e $P' \not\rightarrow$ implica $Q \Longrightarrow^{\tau} Q'$ e $Q' \not\rightarrow$ e $\gamma_M(P', C^{\tau}) = \gamma_M(Q', C^{\tau})$ para toda classe de equivalência C de $\mathcal{R}$,*

onde $P' \not\rightarrow$ denota que não existe nenhuma transição τ a partir de P' (o mesmo para ($Q' \not\rightarrow$, Q').

O algoritmo para fatorar uma IMC preservando comportamento de acordo com os critérios de bissimilaridade fraca apresentado em [Her02] tem complexidade de tempo $O(n(m''_I + m_M))$, onde n é o número de estados, m_M é o número de transições Markovianas e m''_I é o número de transições interativas após o fecho transitivo das transições

internas. Além da minimização dos efeitos do não determinismo, o processo tem o mérito de agregar o espaço de estados, o que é uma vantagem para realizar verificação de modelos.

Caso o comportamento de uma IMC possa ser completamente modelado por uma CTMC, tal CTMC será a saída do algoritmo de [Her02]. Caso não seja, a saída do algoritmo será uma IMC bissimilar à IMC de entrada com um número menor ou igual de transições internas, visto que algumas das transições internas podem ter sido suprimidas com sucesso.

A presença do não determinismo pode afetar a análise estatística de muitas Cadeias de Markov, porém uma ampla gama de modelos IMC pode ser transformado em CTMC através deste processo. Restringindo o modelo se pode obter IMCs que sempre sejam redutíveis a CTMCs – basta impedir que o modelo global tenha, para o mesmo estado, ações de sincronização auto-concorrentes que levem a estados não bissimilares [Hoa85]. Consideramos porém que restringir o modelo a fim de evitar o não determinismo não é uma boa idéia, uma vez que o não determinismo é um ingrediente essencial de sistemas multi-agentes; sua presença reflete que muitos sistemas, quando considerados em um dado nível de abstração, comportam-se não deterministicamente a menos que informações adicionais sejam incluídas no modelo.

Resumimos a seguir o processo de minimização do não determinismo a ser aplicado ao modelo IMC global que definimos para modelar um sistema multi-agentes.

1. O primeiro passo consiste em aplicar o algoritmo de bissimilaridade fraca definido em [Her02].

2. Caso o modelo obtido não contenha transições internas, então obtivemos com sucesso uma CTMC.

3. Caso o modelo obtido ainda contenha transições internas, então o processo Markoviano subjacente está subespecificado, ou seja, a incerteza agregada pelo não determinismo ao modelo não pode ser desprezada. Este fato é uma conseqüência de que ações internas concorrentes, partindo de um mesmo estado, possam levar a comportamentos distintos do sistema.

Refinando a noção de conhecimento

Se é o caso que o modelo de um sistema multi-agentes pode ser representado por (ou reduzido a) uma CTMC, então o comportamento da CTMC obtida representa o comportamento evolutivo da IMC original, baseado nas premissas de progresso maximal (para ditar o comportamento no tempo de transições interativas) e no comportamento estocástico das transições Markovianas. Podemos tirar proveito disso para refinar a noção de conhecimento:

- Estados "instáveis", ou seja, estados de onde se originavam transições internas τ na IMC original foram suprimidos pois não influenciam o consumo do tempo, ou seja, o tempo que o sistema passa nestes estados, ditado pelo progresso maximal, é zero, pois a transição interna é realizada imediatamente.

- Para os estados restantes, que permanecem na CTMCs gerada a partir do modelo original, podemos lançar mão dos processo de análise de CTMC para estimar o comportamento desta no tempo.

Relembramos que CTMCs homogêneas e com estados finitos (que são as que consideramos nesta tese) podem exibir padrões de comportamento ao longo do tempo, que são capturados pela classe de propriedades de estado estacionário (ou equilíbrio estacionário), discutidas na seção 3.3.1. Considerando que a distribuição de probabilidade do equilíbrio estacionário pode ser utilizada para inferir a porcentagem do tempo, na longa jornada, que a CTMC passa em cada estado, esta distribuição pode ser usada por um agente para acreditar que alguns estados sejam mais prováveis que outros dentro de seu conjunto de estados indistinguíveis. Assim, poderemos raciocinar sobre conhecimento e probabilidade, e avaliar fórmulas que expressam coisas do tipo "de acordo com o agente i, a formula ϕ vale com probabilidade de pelo menos b."

Podemos utilizar a probabilidade de equilíbrio estacionário para calcular probabilidades que podemos associar a relação $\sim_i$, da forma apresentada [FH94] e descrita na seção 2.4. Um agente ao adentrar um estado global s considera que o estado global pode ser qualquer um entre os estados indistinguíveis de s, porém considera que dentre o conjunto de indistinguíveis alguns estados sejam mais prováveis que outros.

As probabilidades do equilíbrio estacionário correspondem a probabilidade de uma CTMC estar no estado s em um dado momento ao longo do tempo dado que o estado inicial foi s_0, $\pi_{s_0}(s)$ (definição 3.3.9). Podemos calcular a probabilidade do equilíbrio estacionário para cada um dos estados da CTMC. Caso uma "foto" seja tirada do sistema após este estar em execução por um longo tempo, as probabilidades do equilíbrio estacionário representam a distribuição correspondente a qual seria o estado do sistema na foto. Utilizaremos esta probabilidade para definir o conjunto de probabilidades que um agente atribui aos estados indistinguíveis de s. Apresentamos a seguir a intuição por trás desta definição.

Considere que o estado global do sistema é s. O agente i conhece seu próprio estado local $s[i]$, mas não sabe qual dentre os estados do sistema onde o estado local é o mesmo corresponde ao atual estado global – em outras palavras, não sabe qual dos estados indistinguíveis de s é o estado global corrente, e considera que o estado corrente pode ser qualquer um dos estados s' tal que $s \sim_i s'$. Ao fazer essa consideração o agente está na verdade tentando imaginar como seria uma foto tirada do sistema neste momento, com o adendo de que ele tem já parte do princípio de que o estado atual é compatível com seu estado local. O agente poderia então utilizar as probabilidades do estado estacionário para obter uma distribuição de probabilidades sobre os estados indistinguíveis.

Essa é uma abordagem natural e elegante, porém alguns ajustes são necessários porque o conjunto de estados indistinguíveis do agente advém do modelo IMC original, e as probabilidades estacionárias advém da CTMC subjacente ao modelo. A CTMC subjacente gerada pelo procedimento de [Her02] tem o mérito de permitir agregar o espaço de estados preservando o comportamento evolutivo do sistema, porém isso é atingido à custa de negligenciar a identidade dos estados em favor das classes de equivalência de estados exibindo comportamentos idênticos. O fato é que a CTMC subjacente pode ter um número menor de estados que a IMC original. Porém, devemos lembrar que, se o procedimento efetivamente obteve uma CTMC a partir da IMC original é porque o comportamento evolutivo da segunda pôde ser completamente modelado pela primeira; os estados que foram agregados são representados por suas classes e podemos tirar informação também sobre eles a partir da CTMC.

Se uma CTMC chegou a ser obtida pelo procedimento de [Her02], cada classe de estados gerada pode conter dois tipos de estados:

- *Estados instáveis*, ou seja, aqueles a partir dos quais uma transição interna τ pode ser realizada. Estes são os estados que consideramos, pela premissa do progresso maximal, que não contribuem para a passagem do tempo – o sistema realiza a transição interna τ tão logo entra no estado;

- *Estados estáveis*, que são estados de onde não é possível realizar transições τ, que possuem o mesmo comportamento Markoviano, ou seja, cujo comportamento evolutivo é idêntico.

Uma mesma classe pode conter apenas estados estáveis, ou ainda estados dos dois tipos. Não há classes contento apenas estados instáveis, ou o comportamento de tal classe não teria como ser especificado em uma CTMC.

A premissa do progresso maximal é na verdade o que dita o comportamento dos estados instáveis: o sistema passa por eles de forma imediata, e nenhum tempo é gasto no estado. Daí é natural assumir que em uma foto tirada do sistema a probabilidade de que o estado corrente seja um estado instável é zero, para qualquer estado instável.

Falta ainda considerar o caso onde um estado da CTMC representa uma classe onde mais de um estado estável da IMC original foi agrupado. As probabilidades do equilíbrio estacionário da CTMC representam a probabilidade de que, dado um instante qualquer de tempo, o estado corrente do sistema seja s, para cada s pertencente ao conjunto de estados da CTMC; ou dito de outra forma, a probabilidade do equilíbrio estacionário representa a porcentagem do tempo em que o sistema passa na classe de estados correspondente da IMC original. Relembrando que o algoritmo de [Her02] agrupa em uma mesma classe estados estáveis cujo comportamento evolutivo é idêntico, podemos considerar que os estados estáveis da IMC original pertencentes a uma mesma classe correspondente a um estado da CTMC subjacente são equiprováveis em relação a probabilidade da classe, ou seja, se a probabilidade do equilíbrio estacionário calculada para a classe é p, e a classe agrupa m estados estáveis da IMC original, associaremos a cada um destes estados uma probabilidade de $\frac{p}{m}$.

Assim concluímos o tratamento de todos os tipos de estados que podem ser agrupados em uma classe, e estamos prontos para definir a probabilidade $\mu_{i,s}(s')$ que o agente i associa ao estado s' no estado s.

Seja $S_{\sim_i,s}$ o conjunto de estados indistinguíveis de s para o agente i, ou seja: $s'' \in S_{\sim_i,s}$ se e somente se $s \sim_i s''$. $\mu_{i,s}(s')$ é igual a probabilidade condicional de que o estado global do sistema seja s', dado que o estado global do sistema pertence ao conjunto $S_{\sim_i,s}$. Ambas as duas probabilidades necessárias para calcular $\mu_{i,s}(s')$ podem ser obtidas conforme descrito anteriormente:

Definição 4.3.9 *A probabilidade $\mathfrak{pe}_{s_0,s'}$ de que o estado de uma IMC seja s' em um momento qualquer no tempo, dado que o estado inicial foi s_0 e considerando que $C(s')$ denota a classe da CTMC subjacente à qual s' pertence:*

- *Se s' é um estado instável, a probabilidade é zero: $\mathfrak{pe}_{s_0,s'} = 0$*
- *Se s' é um estado estável, a probabilidade é dada pela probabilidade do equilíbrio estacionário $\pi_{s_0}(C(s'))$ de $C(s')$ na CTMC subjacente, dividido pelo número total $Est(C(s'))$ de estados estáveis pertencentes à classe.*

$$\mathfrak{pe}_{s_0,s'} = \frac{\pi_{s_0}(C(s'))}{Est(C(s'))}$$

Conforme mencionamos, $Est(C(s'))$ é sempre maior do que zero pois uma classe tem necessariamente pelo menos 1 estado estável que a represente.

Definição 4.3.10 *A probabilidade $\mathfrak{ps}_{s_0,S_{\sim_i,s}}$ de que o estado corrente da IMC pertença ao conjunto $S_{\sim_i,s}$ em um momento qualquer do tempo, considerando que o estado inicial foi s_0:*

$$\mathfrak{ps}_{s_0,S_{\sim_i,s}} = \sum_{s'' \in S_{\sim_i,s}} \mathfrak{pe}_{s_0,s''}.$$

Utilizando a regra para cálculo de probabilidade condicional, podemos finalmente concluir que:

$$\mu_{i,s}(s') = \frac{\mathfrak{pe}_{s_0,s'}}{\mathfrak{ps}_{s_0,S_{\sim_i,s}}}.$$

$\mu_{i,s}(s')$ pode ser calculado para todos os estados utilizando tão somente informações contidas no modelo original e a CTMC subjacente, que por sua vez também é obtida a

partir do modelo original. Vale ressaltar que existem técnicas para calcular as probabilidades do equilíbrio estacionário baseadas em resolução de um sistema de equações lineares, conhecidas como as "equações de balanço" [Ste94].

Enriqueceremos o modelo de IMC $\sim$composta com as probabilidades $\mu_{i,s}$ que acabamos de definir:

Definição 4.3.11 *(IMC para Conhecimento Probabilístico)*

Dado um conjunto N de agentes e um conjunto Φ de proposições, uma IMC para Conhecimento Probabilístico $(I)_P = (S, Act, \rightarrowtail, \twoheadrightarrow, P, \sim_1, \sim_2, \cdots, \sim_n, \pi, \mathcal{P})$ é uma IMC $\sim$composta acrescida de:

- *uma atribuição π de valores verdade às primitivas de Φ em cada estado $s \in S$.*
- *uma função de atribuição de probabilidades $\mathcal{P}$ que atribui a cada agente $i \in N$ e estado $s \in S$ um espaço de probabilidades $\mathcal{P}(i,s) = (S_{i,s}, \mathcal{X}_{i,s}, \mu_{i,s})$,*
  onde $S_{i,s} = \mathcal{X}_{i,s} = S_{\sim_i,s}$, e $\mu_{i,s}$ é dado para cada s' de $S_{\sim_i,s}$ por:

$$\mu_{i,s}(s') = \frac{\mathfrak{pe}_{s_0,s'}}{\mathfrak{ps}_{s_0,S_{\sim_i,s}}}.$$

Obtemos assim um modelo expressivo o suficiente para que possamos falar de conhecimento e probabilidade.

Vale mencionar que um modelo com separação entre ações interativas e estocásticas também poderia ter sido adotado para o caso de modelos com tempo discreto (seção 4.2.1), de forma a evitar ou ao menos minimizar a necessidade de adversários para resolver as escolhas não-determinísticas. Porém o modelo MDP é uma opção mais intuitiva para a composição de modelos estocásticos do tipo DTMC, e possui a vantagem de não requerer restrições para que seu comportamento evolutivo possa ser verificado com relação a especificações escritas em PCTL. Caso o tratamento do não determinismo por meio de adversários seja indesejável, é possível lançar mão de modelos Cadeias Interativas de Markov para tempo discreto [Her02], e proceder com os processos de modelagem de forma semelhante ao que apresentamos nesta seção para modelos de tempo contínuo.

4.3.2 Especificação: lógica CSL com operador de conhecimento K–CSL

Conforme discutido na seção 3.3.2 podemos utilizar a lógica CSL para escrever especificações para CTMCs. Queremos adaptar a semântica de CSL para raciocinar sobre modelos IMC, de forma que ao avaliar operadores temporais seja levada em conta a CTMC subjacente.

Para poder raciocinar sobre conhecimento no modelo enriqueceremos a linguagem com operadores modais de conhecimento e de conhecimento probabilístico, respectivamente K_i e w_i, para cada agente i. A linguagem resultante será chamada de K–CSL. Como esperado, as relações $\sim_i$ e as distribuições $\mu_{i,s}$ serão utilizadas para dar semântica aos operadores K_i e w_i.

Sintaxe e Semântica para K–CSL

A sintaxe de K–CSL é tal como dada a seguir:

$\phi ::= true \mid a \mid \neg\phi \mid \phi \wedge \phi \mid K_i\phi \mid a_1 w_i\phi_1 + \cdots + a_m w_i\phi_m \geq b \mid \mathcal{P}_{\bowtie p}[\psi] \mid \mathcal{S}_{\bowtie p}[\phi] \mid \mathcal{E}_{\bowtie c}[\phi]$

$\psi ::= \phi \mid \phi \mathcal{U}^I \phi \mid \phi \mathcal{U} \phi$

onde:

- a é uma proposição atômica,
- $i \in N$, onde N é o conjunto de agentes,
- $a_1 \ldots a_m, b$ são números arbitrários e $m \geq 1$,
- $\bowtie \in \{\leq, <, \geq, >\}$,
- $p \in [0, 1]$,
- $c \in \mathbb{R}^{\geq 0}$ e
- I é um intervalo de $\mathbb{R}^{\geq 0}$.

Fórmulas K–CSL são avaliadas sobre IMCs para Conhecimento Probabilístico. A semântica de fórmulas do tipo $true$, a, $\neg\phi$, $\phi \wedge \phi$, é feita da maneira usual. Fórmulas do tipo $\mathcal{P}_{\bowtie p}[\psi]$, $\mathcal{S}_{\bowtie p}[\phi]$ são avaliadas na CTMC subjacente da mesma forma que na

semântica de CSL. A semântica das modalidades de conhecimento K_i é dada da forma usual, baseada nas relações epistêmicas de acessabilidade $\sim_i$ – conforme apresentado na seção 2.1. A semântica das modalidades de conhecimento probabilístico w_i é dada como apresentado em [FH94], baseada nas distribuições $\mu_{i,s}$ – conforme apresentado na seção 2.4.1.

Definição 4.3.12 *(Relação de Satisfação para fórmulas de K–CSL)*

Seja $(I)_P = (S, Act, \rightarrowtail, \twoheadrightarrow, P, \sim_1, \sim_2, \cdots, \sim_n, \pi, \mathcal{P})$ *uma IMC para conhecimento probabilístico. A relação de satisfação para fórmulas K–CSL é definida da seguinte forma:*

$s \models true$ *para todo* $s \in S$,

$s \models a$ *se e somente se* $a \in \pi(s)$,

$s \models \neg\phi$ *se e somente se* $s \not\models \phi$,

$s \models \phi_1 \wedge \phi_2$ *se e somente se* $s \models \phi_1$ *e* $s \models \phi_2$,

$s \models \mathcal{P}_{\bowtie p}[\psi]$ *se e somente se* $C(s) \models \mathcal{P}_{\bowtie p}[\psi]$ *na CTMC subjacente,*

$s \models \mathcal{S}_{\bowtie p}[\phi]$ *se e somente se* $C(s) \models \mathcal{S}_{\bowtie p}[\phi]$ *na CTMC subjacente,*

$s \models K_i(\phi)$ *se e somente se* *para todo* $s' \in S$ *tal que* $s' \sim_i s$, $s' \models \phi$

$s \models w_i(\phi) \geq b$ *se e somente se* $(\mu_{i,s})(S_{\sim_i,s}(\phi)) \geq b$.

$s \models \quad a_1 w_i(\phi_1) + \cdots + a_k w_i(\phi_k) \geq b$ *se e somente se* $a_1(\mu_{i,s})_*(S_{\sim_i,s}(\phi_1)) + \cdots + a_m(\mu_{i,s})_*(S_{\sim_i,s}(\phi_m)) \geq b$.

Onde $S_{\sim_i,s}(\phi)$ *é o conjunto de estados de* $S_{\sim_i,s}$ *nos quais* ϕ *vale e* $C(s)$ *é a classe da CTMC subjacente a* $(I)_P$ *à qual* s *pertence.*

Considerando agora a CTMC subjacente, a semântica é dada conforme feito para CSL na seção 3.3.2:

$C(s) \models \mathcal{P}_{\bowtie p}[\psi]$ *se e somente se* $p_s(\psi) \bowtie p$,

$C(s) \models \mathcal{S}_{\bowtie p}[\phi]$ *se e somente se* $\sum_{s' \models \phi} \pi_s(s') \bowtie p$,

onde:

$p_s(\psi) \stackrel{def}{=} Prob_s(\{w \in Path_s | w \models \psi\})$,

e para cada caminho $w \in Path_s$*:*

$w \models \phi$ *se e somente se* $w(1) \models \phi$,

$w \models \phi_1 \mathcal{U}^I \phi_2$ *se e somente se* $\exists t \in I$ *tal que* $w@t \models \phi_2 \wedge w@x \models \phi_1 \forall x \in [0, t)$,

$w \models \phi_1 \mathcal{U} \phi_2$ *se e somente se* $\exists k \geq 0$ *tal que* $w(k) \models \phi_2 \wedge w(j) \models \phi_1 \forall j < k$.

4.3.3 Processo para verificação de propriedades epistêmicas

Para verificar se uma fórmula ϕ de K–CSL é satisfeita em um estado s de uma IMC para Conhecimento Probabilístico, utiliza-se o processo recursivo descrito na seção 3.1.3, onde cada operador é tratado em separado.

Para os operadores $\mathcal{P}_{\bowtie p}[\psi]$ e $\mathcal{S}_{\bowtie p}[\phi]$, a verificação é feita na CTMC subjacente, utilizando o mesmo procedimento definido na secção 3.3.3 para operadores de CSL. Para os operadores de conhecimento K_i utiliza-se o procedimento definido na seção 4.1.3. Devemos agora especificar o algoritmo para tratar o operador de conhecimento probabilístico, w_i.

Para verificar se um estado s satisfaz $a_1 w_i \phi_1 + \cdots + a_m w_i \phi_m \geq b$, temos que descobrir se $a_1(\mu_{i,s})_*(S_{\sim_i,s}(\phi_1)) + \cdots + a_m(\mu_{i,s})_*(S_{\sim_i,s}(\phi_m)) \geq b$. Para isso, calculamos individualmente cada termo $a_j(\mu_{i,s})_*(S_{\sim_i,s}(\phi_j))$, para j de 1 até m, somamos os resultados encontrados e avaliamos se este resultado é maior ou igual a b.

Calcular cada termo $a_j(\mu_{i,s})_*(S_{\sim_i,s}(\phi_j))$ implica em calcular $(\mu_{i,s})_*(S_{\sim_i,s}(\phi_j))$. Como $\mu_{i,s}$ é mensurável para cada estado s' de $S_{\sim_i,s}$, o cálculo é bem simples. Primeiro, identifica-se os estados s' de $S_{\sim_i,s}(\phi_j)$, e depois soma-se para cada um deles o valor de $\mu_{i,s}(s')$.

Como estamos lidando com conjuntos finitos de agentes e estados, podemos calcular a priori todos os valores $\mu_{i,s}(s')$ e armazená-los para que sejam diretamente utilizados no processo de verificação.

Algoritmo para calcular $(\mu_{i,s})_*(S_{\sim_i,s}(\phi))$:

Function `Calc`μ (*G, ϕ, i, s*)

G [in parameter] é uma IMC de Conhecimento Probabilístico;
ϕ [in parameter] é a fórmula a ser avaliada;
i [in parameter] é o identificador do agente;
s [in parameter] é um estado;
L é um conjunto de estados, inicialmente vazio;
begin
  $L := S_{\sim_i,s}$
  $(\mu_{i,s})(S_{\sim_i,s}(\phi)) = 0$
  foreach *state* t *in* L **do**
    if $\phi \in label(t)$ **then**
      $(\mu_{i,s})(S_{\sim_i,s}(\phi)) := (\mu_{i,s})(S_{\sim_i,s}(\phi)) + \mu_{i,s}(t)$
    end
  end
  return $(\mu_{i,s})(S_{\sim_i,s}(\phi))$
end

O algoritmo de alto nível para verificação de uma fórmula do tipo $a_1 w_i \phi_1 + \cdots + a_m w_i \phi_m \geq b$ é mostrado a seguir:

Function `CheckW` (*G, ϕ, i, s*)

G [in parameter] é uma IMC de Conhecimento Probabilístico;
ϕ [in parameter] é a fórmula a ser avaliada, da forma $a_1 w_i \phi_1 + \cdots + a_m w_i \phi_m \geq b$;
i [in parameter] é o identificador do agente;
s [in parameter] é um estado;
begin
  $total := 0$
  foreach *term* $a_j(\mu_{i,s})_*(S_{\sim_i,s}(\phi_j))$ *in* ϕ **do**
    $total := total + (a_j *$ `Calc`μ(G, ϕ_j, i, s))
  end
  if $total \geq b$ **then**
    return True
    else
      return False
    end
  end
end

A complexidade do processo de verificação para uma fórmula do tipo $a_1 w_i \phi_1 + \cdots + a_m w_i \phi_m \geq b$ pode ser estimada considerando as complexidades dos algoritmos `Calc`μ e `CheckW`. A complexidade do algoritmo `Calc`μ é $O(|S_{\sim_i,s}|)$, cujo limite superior é $O(|S|)$. A complexidade de `CheckW` é dada pelo número de termos na fórmula $a_1 w_i \phi_1 +$

$\cdots + a_m w_i \phi_m \geq b$ (ou seja, m) multiplicado pela complexidade de $\texttt{Calc}\mu$, pois $\texttt{Calc}\mu$ é executado para cada termo: $O(m * |S|)$. A complexidade de verificação associada ao operador K_i é $O(|S|)$, conforme visto na seção 4.1.3.

Relembramos que a complexidade de tempo para verificar uma fórmula ϕ de CLS sobre uma CTMC $\mathcal{C} = (S, s_0, \mathbf{R})$ é linear em $|\phi|$, polinomial em $|S|$ e linear em $q \cdot t_{max}$, onde $q = max_{s \in S}|\mathbf{Q}(s,s)|$ e t_{max} é o valor máximo encontrado em um parâmetro de um operador $\mathcal{U}$ limitado. Logo, a introdução dos operadores de conhecimento e conhecimento probabilístico não aumenta a complexidade do processo de verificação. Além disso, para $\mathcal{P}_{\bowtie p}[\phi_1 \mathcal{U} \phi_2]$ é requerida a resolução de sistemas de equações lineares de tamanho $|S|$, o mesmo necessário para calcular as probabilidades do equilíbrio estacionário de uma IMC.

4.3.4 Exemplo: Leitor combinado

Consideramos um sistema simples com dois agentes. Dois scaners de imagem que varrem partes diferentes de um mesmo ambiente para que a partir da sincronização das duas leituras se possa obter uma varredura completa do ambiente. Os agentes leitores de imagem, que chamaremos de **Scan1** e **Scan2**, têm funcionamento semelhante: o estado inicial é a realização da leitura, cuja taxa é μ para o Scan1 e λ para o Scan2. Uma vez acabada a leitura, os leitores mudam para o estado 1, onde podem sincronizar os dados lidos.

O modelo para o **leitor** combinado é obtido das IMCs dos dois agentes, fazendo a composição paralela de Scan1 com Scan2 e sincronizando na ação $sinc$, depois abstraíndo esta mesma ação. Executando o procedimento de [Her02] sobre o leitor combinado, obtemos uma CTMC subjacente onde dois estados são agrupados e a transição τ é eliminada.

As IMCs correspondentes aos agentes Scan1, Scan2 e ao leitor combinado, bem como a CTMC subjacente, são mostradas na figura 4.7.

Para verificar propriedades envolvendo conhecimento, devemos calcular $\mathfrak{pe}_{s_0,s}$ para cada estado s do leitor combinado. O estado 11 é instável, logo $\mathfrak{pe}_{s_0,11} = 0$. Os demais estados são estáveis, então devemos calcular para cada um deles a probabilidade do equilíbrio estacionário. Supondo que $\mu = 4$ e $\lambda = 2$, as probabilidades do equilíbrio estacionário da CTMC subjacente são $\frac{1}{3,5}$ para a classe $(00, 11)$, $\frac{2}{3,5}$ para a classe (10), e $\frac{0,5}{3,5}$ para a classe (01). Apesar de que o estado 00 está em uma classe com dois estados

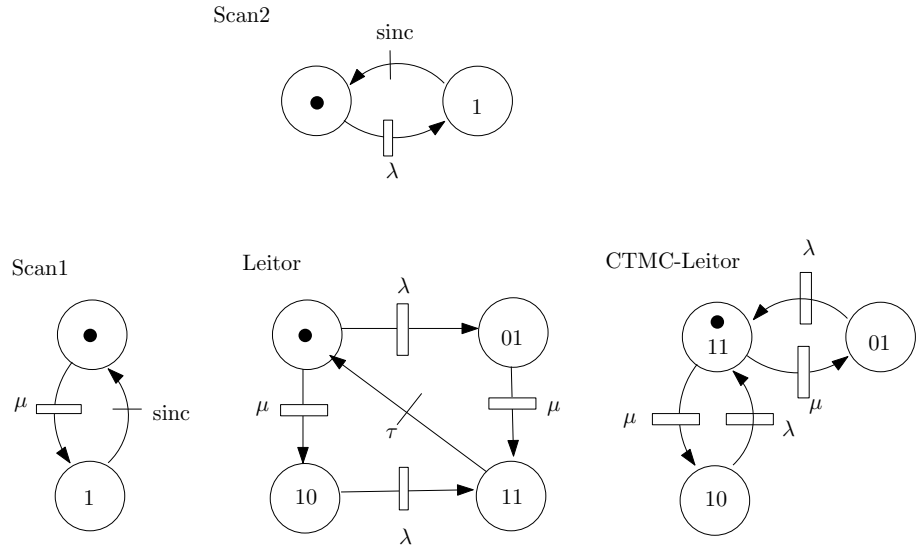


Figura 4.7: Modelagem para o sistema do leitor combinado mostrando as IMCs dos agentes Scan1 e Scan2, a composição seguida de abstração dos IMCs destes dois agentes para a modelagem do leitor, e à direita a CTMC subjacente.

agregados, o estado 11 é instável, então não contribui para o comportamento evolutivo no tempo. Assim sendo, para todos os estados $s \in \{00, 01, 10\}$, $Est(C(s)) = 1$. Logo:

$$\mathfrak{pe}_{00,00} = \frac{1}{3,5}, \;\; \mathfrak{pe}_{00,10} = \frac{2}{3,5}, \;\; \mathfrak{pe}_{00,01} = \frac{0,5}{3,5}, \;\; \mathfrak{pe}_{00,11} = 0$$

.

Os conjuntos de estados indistinguíveis para os agentes Scan1 e Scan2:

- $\sim_{scan1}= \{(00, 01), (10, 11)\}$
- $\sim_{scan2}= \{(00, 10), (01, 11)\}$

As probabilidades $\mathfrak{ps}$ para cada conjunto de estados indistinguíveis:

- $\mathfrak{ps}_{00,scan1(00,01)} = \mathfrak{pe}_{00,00} + \mathfrak{pe}_{00,01} = \frac{1,5}{3,5}$
- $\mathfrak{ps}_{00,scan1(10,11)} = \mathfrak{pe}_{00,10} + \mathfrak{pe}_{00,11} = \frac{2}{3,5}$
- $\mathfrak{ps}_{00,scan2(00,10)} = \mathfrak{pe}_{00,00} + \mathfrak{pe}_{00,10} = \frac{3}{3,5}$
- $\mathfrak{ps}_{00,scan2(01,11)} = \mathfrak{pe}_{00,01} + \mathfrak{pe}_{00,11} = \frac{0,5}{3,5}$

As distribuições μ_i:

- $\mu_{scan1,00}(00) = \mu_{scan1,01}(00) = \mathfrak{pe}_{00,00}/\mathfrak{ps}_{00,scan1(00,01)} = \frac{1}{1,5}$
- $\mu_{scan1,00}(01) = \mu_{scan1,01}(01) = \mathfrak{pe}_{00,01}/\mathfrak{ps}_{00,scan1(00,01)} = \frac{0,5}{1,5}$
- $\mu_{scan1,10}(10) = \mu_{scan1,11}(10) = \mathfrak{pe}_{00,10}/\mathfrak{ps}_{00,scan1(10,11)} = 1$
- $\mu_{scan1,10}(11) = \mu_{scan1,11}(11) = \mathfrak{pe}_{00,11}/\mathfrak{ps}_{00,scan1(10,11)} = 0$
- $\mu_{scan2,00}(00) = \mu_{scan2,10}(00) = \mathfrak{pe}_{00,00}/\mathfrak{ps}_{00,scan2(00,10)} = \frac{1}{3}$
- $\mu_{scan2,00}(10) = \mu_{scan2,10}(10) = \mathfrak{pe}_{00,10}/\mathfrak{ps}_{00,scan2(00,10)} = \frac{2}{3}$
- $\mu_{scan2,01}(01) = \mu_{scan2,11}(01) = \mathfrak{pe}_{00,01}/\mathfrak{ps}_{00,scan2(01,11)} = 1$
- $\mu_{scan2,01}(11) = \mu_{scan2,11}(11) = \mathfrak{pe}_{00,11}/\mathfrak{ps}_{00,scan2(01,11)} = 0$

Os valores encontrados para μ são usados para dar semântica a fórmulas envolvendo o operador de conhecimento probabilístico, como por exemplo:

$$s \models w_{scan2}(lendo1) > 0,5,$$

considerando que $lendo1$ é uma proposição significando que o agente scan1 está realizando a leitura. Esta fórmula é falsa no modelo apresentado: os estados onde $lendo1$ vale são 00 e 01. Uma vez estando em em 00, o agente scan2 considera possíveis os estados 00 e 10, pois $00 \sim_{scan2} 10$. O agente não considera o estado 01 possível a partir de 00, de forma que calculando $(\mu_{scan2,00})_*(\{00, 01\}) = \mu_{scan2,00}(00) = \frac{1}{3}$.

4.3.5 Exemplo: Protocolo do balde transbordante

O "leaky bucket" (balde transbordante) é um mecanismo de controle de acesso utilizado em redes ATM, cujo princípio é configurar o tráfego na rede. Antes de estabelecer uma conexão, os agentes transmissor e receptor negociam com o provedor de acesso alguns parâmetros para definir as características da conexão. Em particular, eles podem negociar uma taxa constante de *buffer* (o balde). Conforme as células de dados chegam na entrada ATM elas entram no balde, o qual é "drenado" com uma "taxa de balde"

específica. Se as células estão chegando mais rapidamente do que o balde é drenado, eventualmente o balde irá transbordar.

Consideramos para este exemplo uma pequena variação do modelo tradicional para o balde transbordante. Consideramos o sistema composto essencialmente por duas filas: uma delas é um *buffer* de células de dados aguardando transmissão, e a outra é uma fila de *tokens*. Para que uma célula de dados seja transmitida ela necessita de um *token*. Se nenhum *token* estiver disponível, a célula tem que aguardar. Assumimos que *tokens* e células possuem tempo de chegada nas filas exponencialmente distribuído, dados respectivamente por λ_T e λ_D. A taxa de serviço, ou seja, a taxa em que um par célula de dado + *token* são retirados das respectivas filas é λ_A.

Assumindo que cada uma das duas filas contém somente duas posições, podemos descrever o sistema como uma composição de IMCs representando o agente transmissor, a fila de céluldas de dados (o balde), o gerador de tokens, a fila de tokens, e o agente ATM responsável por servir ao agente transmissor (e eventualmente entregar as células de dados ao devido receptor, porém esta parte não está representada porque não faz parte do escopo do "leaky bucket"). A figura 4.8 mostra as IMCs para cada uma das partes componentes.

O sistema completo é descrito pela IMC composta da seguinte forma:

- Primeiro, o transmissor é composto com a fila de célula de dados, sincronizando na ação in. A IMC resultante é abstraída para encapsular a ação in.
- Da mesma forma, o gerador de tokens é composto com a fila de tokens também sincronizando na ação in. A IMC resultante é abstraída em in.
- As IMCs obtidas das duas composições anteriores seguidas de abstração são então compostas na ação out.
- A IMC resultante do passo anterior é composta com ATM sincronizando na ação out, que é em seguida encapsulada por abstração.

A IMC resultante do processo descrito possui 72 estados e 244 transições. Aplicando o procedimento de minimização do não determinismo, a IMC resultante possui 23 estados, e não apresenta nenhuma transição interativa. Tal IMC é mostrada na figura 4.9.

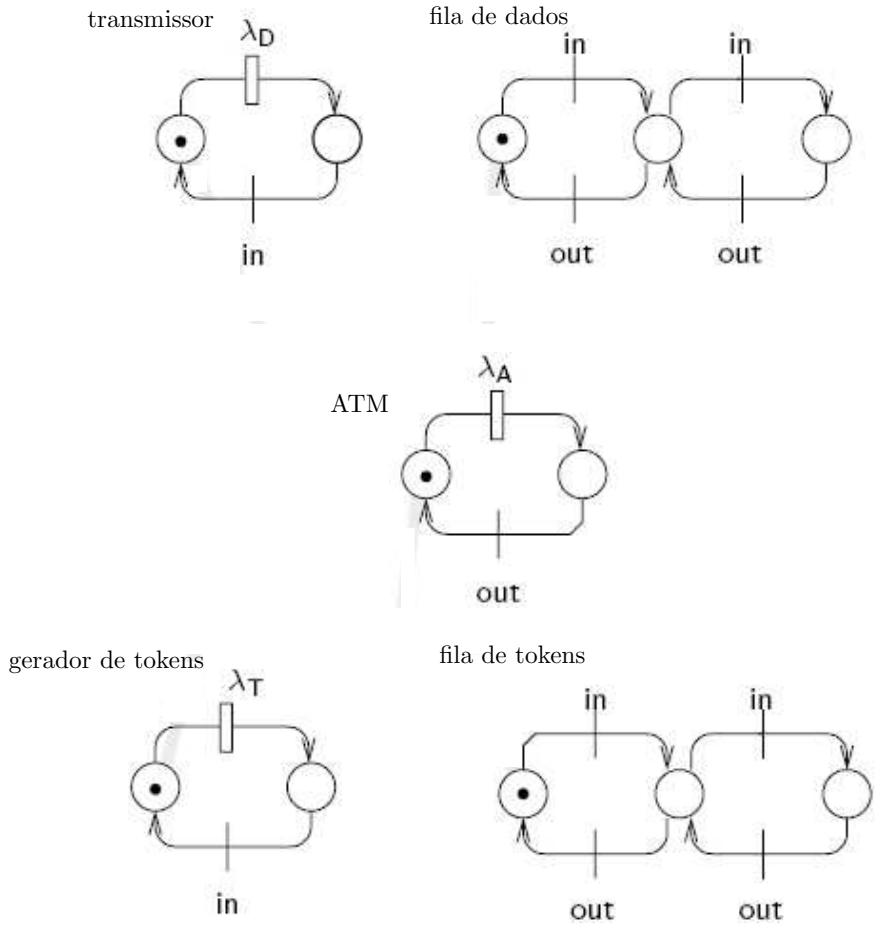


Figura 4.8: Agentes para compor o sistema “Leaky Bucket”.

O processo de verificação pode ser usado sobre a IMC resultante para verificar propriedades como: O agente transmissor sabe que a probabilidade que uma perda de mensagem ocorra nos próximos 30 minutos de transmissão é maior que 89%.

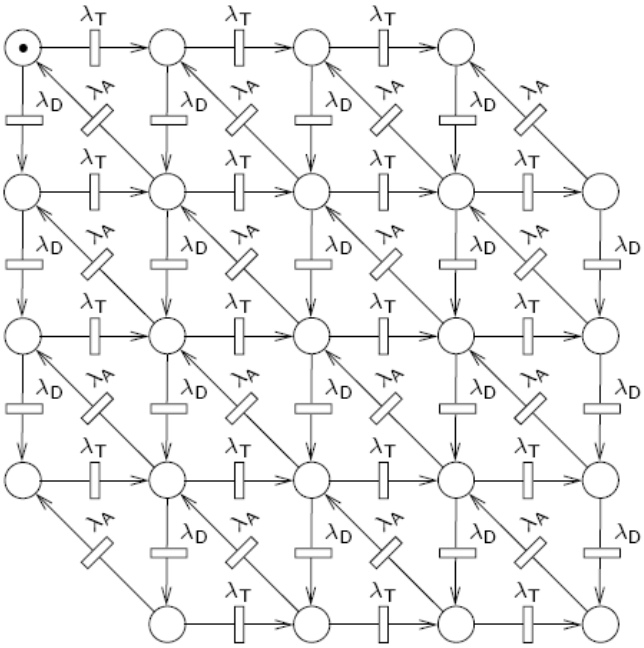


Figura 4.9: Modelo global.

Capítulo 5

Conclusões

O trabalho de pesquisa desta tese concentra-se na área de verificação formal de sistemas multi-agentes. Esta área vem crescendo em importância motivada por resultados positivos no emprego de técnicas de verificação automática em sistemas de larga escala, somada à recente popularidade alcançada pela arquitetura multi-agentes.

O objetivo do trabalho desenvolvido na tese é apresentar estratégias para verificação de sistemas multi-agentes que levem em conta:

- o conhecimento individual de cada agente;
- a composicionalidade da arquitetura do sistema.

Representar o conhecimento é uma forma de refletir no modelo o conceito de que os agentes são independentes e autônomos em seu comportamento. Refletir a composicionalidade da arquitetura no processo de modelagem traz o benefício de reduzir a complexidade desta tarefa, uma vez que o modelo global será obtido diretamente da composição de modelos dos agentes.

Foram investigados em paralelo abordagens para representação de conhecimento e processos de verificação de modelos. Como o processo de verificação de modelos tradicional trabalha com especificações em lógicas modais temporais e muito já está desenvolvido e experimentado sobre este formalismo, decidimos seguir nesta mesma linha para incluir conhecimento nas especificações. Investigamos várias abordagens para representação de conhecimento através de lógicas modais, e usamos as que consideramos mais interessantes como referência para construir a abordagem proposta na tese.

Abordamos três modelos para representação de sistemas concorrentes, discutindo para cada um deles adequações para a representação e verificação de propriedades envolvendo conhecimento:

- modelo não probabilístico de tempo discreto, para o qual existem muitos trabalhos correlatos, fizemos um estudo que revelou ser semelhante ao trabalho contemporâneo [LR06];
- modelo probabilístico de tempo discreto, onde o modelo Processo de Decisão de Markov foi adotado para modelo global, por permitir transições probabilísticas e puramente não determinísticas geradas pela composição de modelos probabilísticos (do tipo Cadeias de Markov de Tempo Discreto). Para análise das propriedades evolutivas desta classe de modelos utilizamos adversários;
- modelo probabilístico de tempo contínuo, onde um modelo também capaz de representar probabilidade de não-determinismo foi adotado, as Cadeias Interativas de Markov. Diferente do modelo MDP, as IMCs permitem em muitos casos analisar o comportamento evolutivo sem a necessidade de adversários, o que permite uma análise transiente e estacionária completa do modelo.

São discutidas para cada classe de modelos como suas características influenciam as relações de conhecimento e como o conhecimento influencia o processo de modelagem.

Elaboramos para cada modelo uma linguagem para raciocinar sobre conhecimento:

- K–CTL para modelos de não probabilísticos de tempo discreto, linguagem CTL acrescida de operadores modais, semelhante à CTL–K de [LR06];
- K–PCTL, para raciocinar sobre conhecimento e tempo em modelos probabilísticos de tempo discreto. A linguagem proposta é o resultado da introdução de operadores modais de conhecimento em PCTL;
- K–CSL, para raciocinar sobre conhecimento, tempo e probabilidade em modelos de tempo contínuo. A linguagem proposta é obtida introduzindo operadores modais de conhecimento e conhecimento probabilístico em CSL.

E construímos os respectivos algoritmos de verificação automática (método explícito) para verificar especificações descritas nas linguagens propostas, sobre os modelos correspondentes. Apesar de algoritmos explícitos de verificação não serem os mais eficientes, são intuitivos e servem de base para construção de algoritmos mais eficientes, como por exemplo algoritmos simbólicos.

Dito de um ponto de vista global, para cada modelo apresentamos um processo envolvendo três fases: modelagem composicional, especificação de propriedades temporais, probabilísticas e epistêmicas, e verificação das especificações sobre os modelos. Para isso, fornecemos:

- um processo composicional para obter os modelos globais do sistema a partir dos modelos dos agentes;
- linguagens para especificar propriedades temporais, probabilísticas (no caso de modelos envolvendo probabilidade) e epistêmicas dos sistemas modelados;
- algoritmos de verificação correspondentes.

Os resultados são muito positivos: não apenas mostramos que é possível realizar a modelagem de sistemas multi-agentes envolvendo conhecimento de forma composicional, como também este processo é completo no sentido de que nenhuma informação é necessária além das informações provenientes dos modelos dos agentes. Isto significa que tal processo pode ser automatizado. Além disso, construir especificações envolvendo propriedades temporais e epistêmicas se mostrou plausível para modelos envolvendo ao mesmo tempo não determinismo e probabilidade, dois tipos distintos de incerteza. Os algoritmos para realizar a verificação das especificações envolvendo conhecimento têm a mesma complexidade de seus correspondentes que trabalham em especificações que não envolvem conhecimento.

Consideramos que seria interessante dar seguimento ao trabalho desenvolvido na tese em três frentes:

- Propor uma implementação para automatizar a construção do modelo global.
- Apresentar um sistema axiomático para as linguagens propostas envolvendo tempo e conhecimento;

- Implementar módulos correspondentes aos algoritmos de verificação para os operadores envolvendo conhecimento para os sistemas existentes de verificação automática, como por exemplo, o PRISM [PRI07].

A implementação do tanto do processo de construção do modelo global quanto dos algoritmos de verificação automática de propriedades epistêmicas proposto nesta tese é completamente viável. Partindo de uma implementação feita em código aberto de um sistema para verificação automática como o PRISM, seria necessário investir em representar e armazenar as relações de acessibilidade no modelo, bem como construir os espaços de probabilidade relacionados ao conhecimento probabilístico; permitir o uso dos operadores de conhecimento e conhecimento probabilístico nas especificações, e interpretá-los de acordo com a semântica descrita; implementar os algoritmos de verificação no mesmo padrão dos já existentes. Consideramos porém que o esforço necessário está além do escopo desta tese.

Referências Bibliográficas

[Ago06] T. Agotnes. Action and knowledge in alternating-time temporal logic. *Synthese*, 146(2), 2006.

[AH90] James Aspnes e Maurice Herlihy. Fast randomized consensus using shared memory. *J. Algorithms*, 11(3):441–461, 1990.

[AHK02] Rajeev Alur, Thomas A. Henzinger e Orna Kupferman. Alternating-time temporal logic. *J. ACM*, 49(5):672–713, 2002.

[ASB95] Adnan Aziz, Vigyan Singhal e Felice Balarin. It usually works: The temporal logic of stochastic systems. In *Proceedings of the 7th International Conference on Computer Aided Verification*, pages 155–165, London, UK, 1995. Springer-Verlag.

[ASSB96] Adnan Aziz, Kumud Sanwal, Vigyan Singhal e Robert K. Brayton. Verifying continuous time markov chains. In Rajeev Alur and Thomas A. Henzinger, editors, *CAV*, volume 1102 of *Lecture Notes in Computer Science*, pages 269–276. Springer, 1996.

[Aum76] Robert J. Aumann. Agreeing to disagree. *Annals of Statistics*, 4(6):1236–1239, 1976.

[Bai98] C. Baier. *On algorithmic verification methods for probabilistic systems*. Habilitation thesis, University of Mannheim, 1998.

[BAMP81] Mordechai Ben-Ari, Zohar Manna e Amir Pnueli. The temporal logic of branching time. In *POPL '81: Proceedings of the 8th ACM SIGPLAN-*

SIGACT symposium on Principles of programming languages, pages 164–176. ACM Press, 1981.

[BdA95] Bianco e de Alfaro. Model checking of probabilistic and nondeterministic systems. *FSTTCS: Foundations of Software Technology and Theoretical Computer Science*, 15, 1995.

[BDR+06] Mario Benevides, Carla Delgado, Ricardo Ribeiro, Luis Lopes e Carlos Pombo. A compositional automata-based approach for model checking multi-agent systems. In *SBMF '06: Simpósio Brasileiro de Métodos Formais*, 2006. to appear in Electronic Notes in Theoretical Computer Science ENTCS.

[BG98] Marco Bernardo e Roberto Gorrieri. A tutorial on empa: a theory of concurrent processes with nondeterminism, priorities, probabilities and time. *Theor. Comput. Sci.*, 202(1-2):1–54, 1998.

[BGS99] Massimo Benerecetti, Fausto Giunchiglia e Luciano Serafini. A model checking algorithm for multiagent systems. In *ATAL '98: Proceedings of the 5th International Workshop on Intelligent Agents V, Agent Theories, Architectures, and Languages*, pages 163–176. Springer-Verlag, 1999.

[BHHK00] Christel Baier, Boudewijn R. Haverkort, Holger Hermanns e Joost-Pieter Katoen. Model checking continuous-time markov chains by transient analysis. In E. Allen Emerson and A. Prasad Sistla, editors, *CAV*, volume 1855 of *Lecture Notes in Computer Science*, pages 358–372. Springer, 2000.

[BHR84] S. D. Brookes, C. A. R. Hoare e A. W. Roscoe. A theory of communicating sequential processes. *J. ACM*, 31(3):560–599, 1984.

[BKH99] Christel Baier, Joost-Pieter Katoen e Holger Hermanns. Approximate symbolic model checking of continuous-time markov chains. In Jos C. M. Baeten and Sjouke Mauw, editors, *CONCUR*, volume 1664 of *Lecture Notes in Computer Science*, pages 146–161. Springer, 1999.

[BS00] D. Brugali e K. Sycara. Towards agent oriented application frameworks. *ACM Computing Surv*, 32(1):21–27, 2000.

[CE81] Edmund M. Clarke e Emerson. Design and synthesis of synchronization skeletons using branching time temporal logic. In *Workshop on Logics of Programs, Lecture Notes in Computer Science 131*, pages 52–71. Springer-Verlag, 1981.

[CY88] C. Courcoubetis e M. Yannakakis. Verifying temporal properties of finite-state probabilistic programs. In IEEE, editor, *29th annual Symposium on Foundations of Computer Science, October 24–26, 1988, White Plains, New York*, pages 338–345, 1988. IEEE catalog no. 88CH2652-6. Computer Society order no. 877.

[CY90] Costas Courcoubetis e Mihalis Yannakakis. Markov decision processes and regular events (extended abstract). In *ICALP '90: Proceedings of the 17th International Colloquium on Automata, Languages and Programming*, pages 336–349, London, UK, 1990. Springer-Verlag.

[CY95] Costas Courcoubetis e Mihalis Yannakakis. The complexity of probabilistic verification. *J. ACM*, 42(4):857–907, 1995.

[dHW03] Wiebe Van der Hoek e Michael Wooldridge. Cooperation, knowledge and time: Alternating-time temporal epistemic logic and its applications. *Studia Logica*, 75(1):125–157, 2003.

[EJT98] Joeri Engelfriet, Catholijn M. Jonker e Jan Treur. Compositional verification of multi-agent systems in temporal multi-epistemic logic. In *Pre-proceedings of the Fifth International Workshop on Agent Theories, Architectures, and Languages, ATAL'98*, pages 91–106. J.P. Mueller, M.P. Singh, A.S. Rao eds., 1998.

[EvdMM98] Kai Engelhardt, Ron van der Meyden e Yoram Moses. Knowledge and the logic of local propositions. In *TARK '98: Proceedings of the 7th con-*

ference on Theoretical aspects of rationality and knowledge, pages 29–41, San Francisco, CA, USA, 1998. Morgan Kaufmann Publishers Inc.

[FG88] Bennett L. Fox e Peter W. Glynn. Computing poisson probabilities. *Commun. ACM*, 31(4):440–445, 1988.

[FH94] Ronald Fagin e Joseph Y. Halpern. Reasoning about knowledge and probability. *J. ACM*, 41(2):340–367, 1994.

[FHM90] R. Fagin, Joseph Y. Halpern e Nimrod Megiddo. A logic for reasoning about probabilities. *Inf. Comput.*, 87(1-2):78–128, 1990.

[FHM95] Ronald Fagin, Joseph Y. Halpern e Y. Moses. *Reasoning about knowledge*. MIT Press, Cambridge, Massachusetts, 1995.

[FLP85] Michael J. Fischer, Nancy A. Lynch e Michael S. Paterson. Impossibility of distributed consensus with one faulty process. *J. ACM*, 32(2):374–382, 1985.

[FZ88] Michael J. Fischer e Lenore D. Zuck. Reasoning about uncertainty in fault-tolerant distributed systems. In *Proceedings of a Symposium on Formal Techniques in Real-Time and Fault-Tolerant Systems*, pages 142–158, New York, NY, USA, 1988. Springer-Verlag New York, Inc.

[GHR93] Norbert Götz, Ulrich Herzog e Michael Rettelbach. Multiprocessor and distributed system design: The integration of functional specification and performance analysis using stochastic process algebras. In Lorenzo Donatiello and Randolph D. Nelson, editors, *Performance/SIGMETRICS Tutorials*, volume 729 of *Lecture Notes in Computer Science*, pages 121–146. Springer, 1993.

[GJ04] V. Goranko e W.J. Jamroga. Comparing semantics of logics for multi-agent systems. *Synthese*, 139(2):241–280, 2004.

[Hal91] Joseph Y. Halpern. Knowledge and probability in distributed systems: abstract. In *TAPSOFT '91: Proceedings of the international joint conference*

on theory and practice of software development on Advances in distributed computing (ADC) and colloquium on combining paradigms for software development (CCPSD): Vol. 2, pages 50–54, New York, NY, USA, 1991. Springer-Verlag New York, Inc.

[HC96] G. E. Hughes e M. J. Cresswell. *A New Introduction to Modal Logic*. Routledge, September 1996.

[Her01] Holger Hermanns. Construction and verification of performance and reliability models. *Bulletin of the EATCS*, 74:135–153, 2001.

[Her02] Holger Hermanns. *Interactive Markov Chains : The Quest for Quantified Quality*, volume 2428 of *Lecture Notes in Computer Science*. Springer Berlin // Heidelberg, 2002.

[Hil96] Jane Hillston. *A compositional approach to performance modelling*. Cambridge University Press, New York, NY, USA, 1996.

[Hin62] Jaakko Hintikka. *Knowledge and belief*. Cornell University Press, New York, 1962.

[HJ94] Hans Hansson e Bengt Jonsson. A logic for reasoning about time and reliability. *Formal Aspects of Computing*, 6(5):512–535, 1994.

[HM84] Joseph Y. Halpern e Yoram Moses. Knowledge and common knowledge in a distributed environment. In *PODC '84: Proceedings of the third annual ACM symposium on Principles of distributed computing*, pages 50–61, New York, NY, USA, 1984. ACM Press.

[HMT88] Joseph Halpern, Yjoram Moses e Mark Tuttle. A knowledge-based analysis of zero knowledge. In *STOC '88: Proceedings of the twentieth annual ACM symposium on Theory of computing*, pages 132–147, New York, NY, USA, 1988. ACM Press.

[Hoa85] C. A. R. Hoare. *Communicating Sequential Processes*. Prentice Hall, 1985.

[Hol91] Gerard J. Holzmann. *Design and validation of computer protocols*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1991.

[Hol04] Gerard J. Holzmann. *The Spin Model Checker - Primer and Reference Manual*. Addison-Wesley, 2004.

[HR95] Matthew Hennessy e Tim Regan. A process algebra for timed systems. *Inf. Comput.*, 117(2):221–239, 1995.

[HR99] Holger Hermanns e Michael Rettelbach. Syntax, semantics, equivalences and axioms for mtipp. In *2nd Workshop on Process Algebras and Performance Modelling, PAPM 94, Regensberg (Germany)*, volume 27 of *Arbeitsberichte des Instituts für Mathematische Machinen und Datenverarbeitung*, pages 71–88. Universität Erlangen-Nürnberg, 1999.

[HV89] Joseph Y. Halpern e Moshe Y. Vardi. The complexity of reasoning about knowledge and time. i. lower bounds. *J. Comput. Syst. Sci.*, 38(1):195–237, 1989.

[Jon03] G. Jonker. Feasible strategies in alternating-time temporal epistemic logic. Master's thesis, University of Utrech, The Netherlands, 2003.

[JvdH04] Wojciech Jamroga e Wiebe van der Hoek. Agents that know how to play. *Fundam. Inf.*, 63(2-3):185–219, 2004.

[KKNP01] Joost-Pieter Katoen, Marta Z. Kwiatkowska, Gethin Norman e David Parker. Faster and symbolic ctmc model checking. In Luca de Alfaro and Stephen Gilmore, editors, *PAPM-PROBMIV*, volume 2165 of *Lecture Notes in Computer Science*, pages 23–38. Springer, 2001.

[KLP04] Magdalena Kacprzak, Alessio Lomuscio e Wojciech Penczek. Verification of multiagent systems via unbounded model checking. In *AAMAS '04: Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 638–645, Washington, DC, USA, 2004. IEEE Computer Society.

[Kri59] Saul A. Kripke. A completeness theorem in modal logic. *Journal of Symbolic Logic*, 24:1–15, 1959.

[Leh84] Daniel Lehmann. Knowledge, common knowledge and related puzzles (extended summary). In *PODC '84: Proceedings of the third annual ACM symposium on Principles of distributed computing*, pages 62–67. ACM Press, 1984.

[LMWF94] Nancy Lynch, Michael Merritt, William Weihl e Alan Fekete. *Atomic Transactions*. Morgan Kaufmann Publishers, San Mateo, CA, 1994.

[LR06] Alessio Lomuscio e Franco Raimondi. Model checking knowledge, strategies, and games in multi-agent systems. In *AAMAS '06: Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems*, pages 161–168, New York, NY, USA, 2006. ACM Press.

[LS82] D. Lehmann e S. Shelah. Reasoning with time and chance. *Information and Control*, 53:165–198, 1982.

[McM02] Kenneth L. McMillan. Applying sat methods in unbounded symbolic model checking. In *CAV '02: Proceedings of the 14th International Conference on Computer Aided Verification*, pages 250–264, London, UK, 2002. Springer-Verlag.

[MDH85] Yoram Moses, Danny Dolev e Joseph Y. Halpern. Cheating husbands and other stories (preliminary version): a case study of knowledge, action, and communication. In *PODC '85: Proceedings of the fourth annual ACM symposium on Principles of distributed computing*, pages 215–223, New York, NY, USA, 1985. ACM Press.

[Mil89] R. Milner. *Communication and concurrency*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1989.

[MM04] Annabelle McIver e Carroll Morgan. *Abstraction, Refinement And Proof For Probabilistic Systems (Monographs in Computer Science)*. SpringerVerlag, 2004.

[Moo85] R. C. Moore. *Formal Theories of the Commonsense World*, chapter A Formal Theory of Knowledge and Action, pages 319–358. Ablex, Norwood, NJ, 1985.

[Nil86] Nils J. Nilsson. Probabilistic logic. *Artif. Intell.*, 28(1):71–88, 1986.

[NS92] Xavier Nicollin e Joseph Sifakis. An overview and synthesis on timed process algebras. In *CAV '91: Proceedings of the 3rd International Workshop on Computer Aided Verification*, pages 376–398, London, UK, 1992. Springer-Verlag.

[PCG00] Doron A. Peled, Edmund M. Clarke e Orna Grumberg. *Model Checking*. MIT Press, Cambridge, Massachusetts, 2000.

[PL03] Wojciech Penczek e Alessio Lomuscio. Verifying epistemic properties of multi-agent systems via bounded model checking. *Fundam. Inf.*, 55(2):167–185, 2003.

[PRI07] Prism website, 2007. http://www.prismmodelchecker.org/, visitado em 20 de Outubro de 2007.

[PT92] Prakash Panangaden e Kim Taylor. Concurrent common knowledge: Defining agreement for asynchronous systems. *Distributed Computing*, 6(2):73–93, 1992.

[Rab82] Michael O. Rabin. N-process mutual exclusion with bounded waiting by 4 log 2 n-valued shared variable. *J. Comput. Syst. Sci.*, 25(1):66–75, 1982.

[RKNP04] J. Rutten, M. Kwiatkowska, G. Norman e D. Parker. *Mathematical Techniques for Analyzing Concurrent and Probabilistic Systems*, volume 23 of *CRM Monograph Series*. P. Panangaden and F. van Breugel (eds.), 2004.

[SO03] M. Wooldridge S. Otterloo, W. van der Hoek. Knowledge as strategic ability. *ENTCS*, 85(2):1–23, 2003.

[Ste94] William J. Stewart. *Introduction to the Numerical Solution of Markov Chains*. Princeton University Press, Princeton, NJ, 1994.

[Var85] Moshe Y. Vardi. Automatic verification of probabilistic concurrent finite-state programs. In *FOCS*, pages 327–338. IEEE, 1985.

[vdHW02a] Wiebe van der Hoek e Michael Wooldridge. Model checking knowledge and time. In *Proceedings of the 9th International SPIN Workshop on Model Checking of Software*, pages 95–111, London, UK, 2002. Springer-Verlag.

[vdHW02b] Wiebe van der Hoek e Michael Wooldridge. Tractable multiagent planning for epistemic goals. In *AAMAS '02: Proceedings of the first international joint conference on Autonomous agents and multiagent systems*, pages 1167–1174, New York, NY, USA, 2002. ACM Press.

[vdMS99] Ron van der Meyden e Nikolay V. Shilov. Model checking knowledge and time in systems with perfect recall (extended abstract). In *Foundations of Software Technology and Theoretical Computer Science*, pages 432–445, 1999.

[Yi91] Wang Yi. Ccs + time = an interleaving model for real time systems. In *Proceedings of the 18th international colloquium on Automata, languages and programming*, pages 217–228, New York, NY, USA, 1991. Springer-Verlag New York, Inc.