COPPE/UFRJ

DESENVOLVIMENTO DE UMA ARQUITETURA DE NAVEGAÇÃO DELIBERATIVA PARA ROBÔS MÓVEIS UTILIZANDO A TEORIA DE CONTROLE SUPERVISÓRIO

Lucas Molina

Dissertação de Mestrado apresentada ao Programa de Pós-graduação em Engenharia Elétrica, COPPE, da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Mestre em Engenharia Elétrica.

Orientadores: João Carlos dos Santos Basílio
Eduardo Oliveira Freire

Rio de Janeiro
Março de 2010

DESENVOLVIMENTO DE UMA ARQUITETURA DE NAVEGAÇÃO DELIBERATIVA PARA ROBÔS MÓVEIS UTILIZANDO A TEORIA DE CONTROLE SUPERVISÓRIO

Lucas Molina

DISSERTAÇÃO SUBMETIDA AO CORPO DOCENTE DO INSTITUTO ALBERTO LUIZ COIMBRA DE PÓS-GRADUAÇÃO E PESQUISA DE ENGENHARIA (COPPE) DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS EM ENGENHARIA ELÉTRICA.

Examinada por:

Prof. João Carlos dos Santos Basílio, Ph. D.

Prof. Eduardo Oliveira Freire, D. Sc.

Prof. Liu Hsu, Dr. d'État

Prof. Antonio Eduardo Carrilho da Cunha, D. Eng.

RIO DE JANEIRO, RJ – BRASIL
MARÇO DE 2010

Molina, Lucas

Desenvolvimento de uma arquitetura de navegação deliberativa para robôs móveis utilizando a teoria de controle supervisório/Lucas Molina. – Rio de Janeiro: UFRJ/COPPE, 2010.

XIV, 91 p.: il.; 29, 7cm.

Orientadores: João Carlos dos Santos Basílio
Eduardo Oliveira Freire

Dissertação (mestrado) – UFRJ/COPPE/Programa de Engenharia Elétrica, 2010.

Referências Bibliográficas: p. 87 – 91.

1. controle supervisório. 2. navegação de robôs móveis. 3. planejamento de trajetória. I. Basílio, João Carlos dos Santos *et al.*. II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia Elétrica. III. Título.

A meu pai Heraldo Molina (in memoriam), minha mãe Maria Imaculada Ferreira Molina e minha namorada Christiane Raulino Almeida.

Agradecimentos

Gostaria de agradecer primeiramente a Deus e a todos os meus Orixás pela luz e proteção que sempre me deram, sem a qual eu não poderia estar aqui concluindo esse trabalho.

Ao meu pai, Heraldo Molina (*in memoriam*), que sempre me incentivou e me apoiou nas minhas escolhas, me ensinando a aprender com os acertos e com os erros cometidos. Muito obrigado por tudo.

À minha mãe, Maria Imaculada Ferreira Molina, e aos meus irmãos, Raquel Molina e Henrique Molina, por todo apoio que me deram e por terem me acolhido após anos afastado do convívio em família. Estarmos juntos novamente é muito importante para mim.

À minha namorada, Christiane Raulino Almeida, que me ajudou a suportar a saudade quando estávamos distantes, que teve paciência com a minha ausência quando estávamos juntos e que me ajuda a ser uma pessoa melhor a cada dia que passamos juntos. Te amo.

Aos meus sogros, Sérgio Luiz Vieira Almeida e Sarah Maria Raulino Almeida, que me acolheram em sua casa todas as vezes que estive em Aracaju, sempre me tratando como se trata um filho, me ajudando a superar um dos momentos mais difíceis da minha vida. Muito obrigado.

Ao meu grande amigo Elyson Ádan Nunes Carvalho, que sempre me inspirou desde os meus primeiros passos como pesquisador. Um amigo presente em alguns dos momentos mais difíceis de toda a minha vida. Uma amizade para a vida inteira.

Ao meu orientador, João Carlos dos Santos Basílio, que confiou em mim, que se dispôs a me orientar em um trabalho associando a sua linha de pesquisa à robótica móvel, que compreendeu a minha ausência em diversas ocasiões e que, por muitas vezes, demonstrou por mim a consideração e compreensão que só se pode esperar de um amigo, mesmo me conhecendo há tão pouco tempo. Meus sinceros agradecimentos.

Ao meu orientador, Eduardo Oliveira Freire, que me ensinou a amar a robótica e que acreditou em mim quando me levou para o seu grupo de pesquisa, ainda na graduação, onde eu descobri a minha vocação e comecei a caminhar em busca de um sonho: um dia poder trabalhar com aquilo que eu amo. Muito obrigado.

Ao meu amigo e colega de quarto, Gabriel Matos Araújo, que se tornou um irmão pra mim e para todos da minha família, dividindo todos os momentos comigo durante esses dois anos de trabalho.

A Jugurta Rosa Montalvão Filho, que me inspirou a sair de casa em busca de uma melhor formação em uma instituição de excelência, pelas conversas esclarecedoras e pelo incentivo nos momentos de maior descrença. Você estava certo! Eu aprendi muito e em todos os aspectos.

A todos os colegas de laboratório, em especial a Thiago Cerqueira de Jesus (Baiano), que esteve comigo desde o período de matérias, e a Lilian Kawakami Carvalho, sempre dispostos a ajudar durante todas as etapas desse trabalho.

A todos que deixei em Aracaju e que me fizeram tanta falta nesse período, a ponto de quase desistir, mas que em todos os momentos me incentivaram a continuar.

Por fim, gostaria de agradecer à Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - CAPES, pelo suporte financeiro, sem o qual seria impossível concluir este trabalho.

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

DESENVOLVIMENTO DE UMA ARQUITETURA DE NAVEGAÇÃO DELIBERATIVA PARA ROBÔS MÓVEIS UTILIZANDO A TEORIA DE CONTROLE SUPERVISÓRIO

Lucas Molina

Março/2010

Orientadores: João Carlos dos Santos Basílio
Eduardo Oliveira Freire

Programa: Engenharia Elétrica

O aumento da complexidade das tarefas realizadas pelos robôs móveis fez crescer a demanda por sistemas mais complexos de navegação. Nesses sistemas não basta apenas concluir uma tarefa, mas, principalmente, concluí-la satisfazendo diversas especificações de desempenho. Tais requisitos demandam um sistema de controle independente capaz de satisfazer a mais de um objetivo (comportamento) ao mesmo tempo. A modelagem de comportamentos utilizando elementos da teoria de sistemas a eventos discretos surge, então, como uma alternativa para se levar em conta as interações entre os diversos comportamentos de um sistema, uma vez que permite a inclusão de novos comportamentos sem que, para isso, seja necessário alterar os demais comportamentos já modelados. Seguindo essa ideia, esse trabalho apresenta o desenvolvimento e a implementação de uma arquitetura de navegação predominantemente deliberativa, utilizando coordenação de comportamentos modelados utilizando-se autômatos e um critério de planejamento baseado em uma medida de linguagem (μ). O sistema de planejamento de trajetória modela todos os possíveis movimentos do robô, sendo formado, basicamente, por um autômato de planejamento, que modela o ambiente de navegação e as ações do robô nesse ambiente. Considerando as trajetórias que levam o robô de um ponto inicial a um ponto de destino como sendo sublinguagens da linguagem gerada pelo autômato de planejamento, é possível utilizar o parâmetro μ como medida de desempenho dessas trajetórias, possibilitando, assim, a escolha daquela que apresentar o melhor desempenho. Exercícios de simulação realizados utilizando-se o *software* MobileSim, disponibilizado pela fabricante do robô P3-DX, comprovam a eficácia da arquitetura de navegação proposta nesse trabalho.

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

DEVELOPMENT OF A DELIBERATIVE NAVIGATION ARCHITECTURE FOR MOBILE ROBOTS USING SUPERVISORY CONTROL THEORY

Lucas Molina

March/2010

Advisors: João Carlos dos Santos Basílio
Eduardo Oliveira Freire

Department: Electrical Engineering

The increase in the complexity of the tasks performed by mobile robots has grown the demand for more complex navigation systems. In these systems, it is not enough to perform a task but, also, to perform it satisfying several performance specifications. Such specifications require an independent control system that is able to achieve several objectives (behaviors) simultaneously. Behavior modeling using the theory of discrete event systems has arisen as an alternative to take into account all the interaction between the different required behaviors, since it allows the inclusion of new modeled behaviors without the need to change the other already modeled behaviors. Following this idea, this work presentes a development and implementation of a navigation architecture predominantly deliberative using behavior coordination by using automata and a planning criterion based on a measure of language (μ). The trajectory planning system models all robot possible moves and is basically formed of a planner automaton that models the navigation environment and the robot actions in this environment. By taking into account all possible robot trajectories from an inicial to a final point as the sub-language of the language generated by the planner automaton, it is possible to use parameter μ as a measure of performance for these trajectories so as to choose the best trajectory according to the adopted criterion. Simulation exercises carried out using MobileSim software, made available by P3-DX robot manufacturer, demonstrate the efficiency of the proposed navigation architecture.

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

Segurança, velocidade de execução e repetibilidade são características que levam a robótica a ser cada vez mais utilizada para execução de tarefas que antes eram feitas apenas por seres humanos. Robôs são hoje facilmente encontrados executando tarefas que combinam capacidade de repetição e necessidade de torque elevado, como por exemplo, em linhas de montagem, ou realizando procedimentos em ambientes que oferecem risco ao homem, como por exemplo, inspeções de tanques de armazenamento de substâncias nocivas à saúde humana, ou mesmo trabalhando em situações em que o homem dificilmente conseguiria sobreviver sozinho, como por exemplo, em missões de exploração de longa duração em outros planetas.

Robôs não sentem sede ou fome, tampouco são capazes de julgar o risco das tarefas às quais são submetidos. No entanto, essa ausência de "personalidade" limita o comportamento desses sistemas a agir e reagir apenas em situações para as quais eles tenham sido programados. Muitas vezes um robô móvel precisa navegar em ambientes dinâmicos, onde a sua capacidade de executar uma tarefa está intimamente associada à sua capacidade de reagir a estímulos externos, desconhecidos, de forma coerente. Essa habilidade pode ser definida de diferentes formas, conhecidas como *mecanismos de seleção de atitude* (MSA), ou simplesmente *arquiteturas de navegação*, que definem que atitude o robô deve tomar a cada momento.

Diversos tipos de arquiteturas de navegação vêm sendo desenvolvidas ao longo dos anos buscando dar maior autonomia e flexibilidade ao robô móvel no tocante à sua capacidade de locomoção. Paolo Pirjanian em [1] apresentou uma ampla revisão bibliográfica sobre os trabalhos desenvolvidos na área de navegação, classificando as arquiteturas apresentadas em deliberativas, que são caracterizadas pela busca da otimalidade, reativas, aquelas caracterizadas pela busca da robustez, e híbridas, que equilibram característica das outras duas escolas.

Com o passar do tempo, a evolução da complexidade das tarefas as quais um robô é submetido aumentou muito e com ela, cresceu a demanda por sistemas de navegação mais complexos, em que a preocupação não é apenas simplesmente con-

cluir uma tarefa e sim concluí-la atendendo diversas especificações que, por si só, demandam um sistema de controle independente.

Para atender mais de um objetivo ao mesmo tempo, surgiram as teorias de composição de comportamentos, nas quais sistemas de controle independentes são desenvolvidos para alcançar diferentes objetivos (comportamentos) mais simples de uma tarefa. Posteriormente, é feita uma composição desses comportamentos, formando um sistema de navegação mais complexo. Ronald C. Arkin, em seu livro *Behavior-Based Robotics* [2], apresentou uma ampla revisão de diferentes formas de composição de comportamentos desenvolvidas até aquele momento em robótica móvel, assim como fez Paolo Pirjanian em [3], que apresentou uma classificação qualitativa dos sistemas de navegação baseados em comportamento, separando-os em sistemas cooperativos, nos quais mais de um comportamento define as ações do robô ao mesmo tempo, e competitivos, em que apenas um comportamento por vez controla os movimentos do robô.

A elaboração de um sistema de composição de comportamentos cooperativos nem sempre é uma tarefa fácil. Assim sendo, diversos pesquisadores buscaram em áreas afins, técnicas e formalismos que possibilitassem a realização da composição de comportamentos de forma mais simples e bem fundamentada. Nesse contexto, a modelagem de comportamentos utilizando a teoria de sistemas a eventos discretos (SED) [4] surge como uma ferramenta que possibilita modelar a interação entre os diversos comportamentos de um sistema de forma simples, permitindo ainda a inclusão de novos comportamentos no sistema, sem que para isso seja necessária qualquer alteração nos demais comportamentos já modelados. Apesar de existirem trabalhos que utilizam a modelagem em redes de Petri para caracterizar os comportamentos de um robô móvel, como em [5], o formalismo mais utilizado pelos pesquisadores da área de navegação é o autômato. Kosecká et al. [6, 7] utilizaram formalmente a teoria de autômatos para demonstrar a viabilidade da utilização dessa teoria na modelagem, análise e síntese de comportamentos baseados em visão computacional aplicados à navegação de um robô móvel bem como para sincronizar esses comportamentos. Após a comprovação da viabilidade da aplicação de SED para tarefas de navegação, foi apresentado em [8] um estudo mais detalhado a respeito da controlabilidade e observabilidade de comportamentos de navegação baseados em autômatos. Em [9, 10] foram realizados experimentos de composição de autômatos, na tentativa de desenvolver novas operações que caracterizassem, da melhor forma possível, a interação entre diferentes comportamentos buscando uma formalização do processo de construção dos comportamentos independentes. Em [11], a teoria de controle supervisório [12, 13] foi considerada na construção e composição de diferentes comportamentos de navegação, possibilitando, assim, o uso de ferramentas de verificação tradicionalmente utilizadas em supervisores. Com a evolução da teoria de controle

supervisório surgiram diferentes métodos de desenvolvimento de um controlador e, com eles, cresceu a busca pela otimalidade dos sistemas controlados. Asok Ray e seus co-autores em [14–16], formalizaram uma medida de linguagem quantitativa (μ), capaz de avaliar o desempenho de sistemas controlados. Essa medida foi utilizada em [17] para definir uma maneira de selecionar os diferentes comportamentos de um sistema de navegação utilizando o conceito de medida de linguagem. Em [18–20], a medida de linguagem μ foi utilizada juntamente com a teoria de sistemas supervisores para o desenvolvimento de um sistema de navegação de robôs móveis, em que o autômato que modela a seleção dos comportamentos foi otimizado segundo o critério de medida de linguagem μ.

Neste trabalho será apresentado o desenvolvimento de uma arquitetura de navegação deliberativa para robôs móveis utilizando a teoria de controle supervisório. O sistema proposto é formado por dois grandes blocos: o sistema supervisor e o sistema de planejamento. No sistema de planejamento será construído um autômato de planejamento P. Nesse autômato, cada estado representa uma posição do robô no ambiente e os eventos representam as ações de movimento do robô. O conceito de medida de linguagem μ será modificado para funcionar como um critério de planejamento, possibilitando assim a escolha da melhor trajetória que leve o robô de um estado (posição inicial) a outro (posição de destino). Uma vez determinada a sequência de eventos a ser executada pelo robô, cabe ao sistema supervisor garantir a execução dessa sequência de forma segura, permitindo o replanejamento da tarefa de navegação, caso um obstáculo venha a ser encontrado pelo robô.

Este trabalho está estruturado da seguinte forma. No capítulo 2 será apresentada uma visão geral sobre arquiteturas de navegação e dos tipos de arquiteturas utilizadas na atualidade, bem como uma classificação qualitativa dessas arquiteturas. No capítulo 3 será feita a fundamentação teórica de sistemas a eventos discretos, necessária para o desenvolvimento deste trabalho. No capítulo 4 serão apresentados os conceitos de controle supervisório utilizados neste trabalho e a formulação da medida de linguagem μ, que será utilizada para avaliar o desempenho de uma linguagem e suas sublinguagens. No capítulo 5 será apresentada a descrição da arquitetura de navegação aqui proposta, sua formulação e os resultados obtidos com essa arquitetura. Finalmente, no capítulo 6 serão apresentadas as conclusões e as sugestões de trabalhos futuros.

Capítulo 2

Arquiteturas de navegação

A navegação de um robô móvel está associada à decisão a respeito de qual ação deve ser executada no instante seguinte. Essa decisão é de fundamental importância, sendo esse problema conhecido como Problema de Seleção de Atitude (PSA) [1]. Os esquemas ou arquiteturas de controle que são utilizados para solucionar este tipo de problema recebem o nome de Mecansismos de Seleção de Atitude (MSA). Tyrrel [21] apresenta a seguinte visão biológica do ASP:

> *"Esse é o problema geral de um animal decidir suas ações de modo a maximizar suas expectativas de sobrevivência e reprodução, ajudando os da sua espécie a fazer o mesmo."*

Essa interpretação é reformulada em [1] utilizando conceitos da teoria de decisão Bayesiana, que defende que os agentes atuem de forma racional ou ótima. O objetivo é dar a ela maior abrangência, adequando-a a casos mais gerais, como por exemplo os agentes móveis. Entretanto, Maes [22] mostra que, devido a condições desfavoráveis como complexidade do ambiente e imprevisibilidade, aliadas às limitações do agente móvel, os MSA não podem ser completamente racionais (ótimos), limitando-se a serem apropriados ou satisfatórios para um dado objetivo. A definição de PSA mais compatível com agentes móveis é apresentada por Maes [22]:

> *"Como um agente pode selecionar a ação seguinte da forma mais apropriada, ou relevante, em um determinado momento, quando se depara com uma situação específica?"*

Nas últimas décadas, diversos MSA foram propostos por pesquisadores de áreas distintas para solucionar problemas específicos dessas áreas, entre as quais destacam-se: inteligência artificial, modelagem de comportamento de agentes biológicos, realidade virtual, agentes de softwares cooperativos e robótica. Nessa última é comum referenciar-se aos MSA como arquiteturas de navegação, termo que será bastante utilizado a partir de agora.

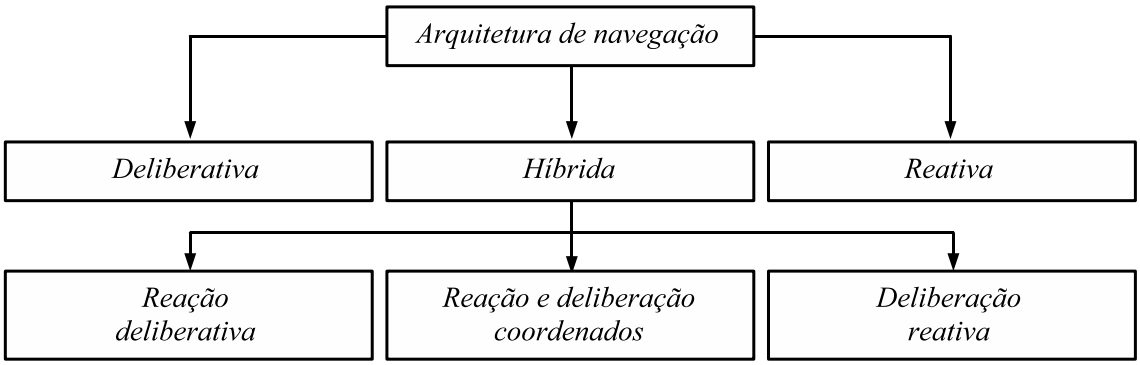


Figura 2.1: Classificação das arquiteturas de navegação, segundo Pirjanian [1].

Esse capítulo está organizado da seguinte forma. Na seção 2.1 é feita uma classificação qualitativa das arquiteturas de navegação existentes na literatura em arquiteturas deliberativas, reativas ou híbridas. A seguir, na seção 2.2 são apresentadas as arquiteturas baseadas em comportamento, um tipo específico de arquiteturas de navegação que se utiliza da composição de comportamentos simples para produzir comportamentos mais complexos, sendo este o tipo de arquitetura que será desenvolvido neste trabalho.

2.1 Classificação qualitativa das arquiteturas de navegação

Com o crescente número de arquiteturas de navegação desenvolvidos na área de robótica móvel, uma classificação qualitativa dos MSA é inevitável, embora não exista unanimidade na comunidade científica atuante a esse respeito. Diversas classificações já foram apresentadas, sendo algumas mais adequadas, e outras até mesmo equivocadas. Uma das classificações mais aceitas pela comunidade científica é aquela apresentada em [1] e aqui ilustrada na figura 2.1. Através de uma análise rápida da figura 2.1, é possível observar que esta classificação sugere uma total independência entre as classes apresentadas (deliberativa, reativa e híbrida). Entretanto, uma rápida avaliação das arquitetuas de navegação comumente encontradas na literatura as colocaria, em sua maior parte, na classe das arquiteturas híbridas. Isso ocorre por que a maioria dos arquiteturas de navegação, carregam características reativas e deliberativas, mas nem sempre em proporções suficientes para serem consideradas híbridas [2].

A classificação que parece a mais completa e abrangente é aquela apresentada por Ronald C. Arkin em [2]. Nessa classificação, as diversas arquiteturas já propostas são agrupadas qualitativamente através de um espectro contínuo que varia desde as

Figura 2.2: Espectro de classificação das arquiteturas de navegação, segundo Arkin [2].

arquiteturas puramente deliberativas ou cognitivas, até as arquiteturas puramente reativas (também conhecidas como reflexivas) de uma forma suave, sem separações bem estabelecidas. Na figura 2.2 é apresentado o espectro de classificação proposto inicialmente em [2]. Conforme pode ser visto na figura 2.2, à medida que se desloca ao longo do espectro, desde as arquiteturas deliberativas em direção às arquiteturas reativas, verifica-se um aumento no grau de acoplamento sensor-atuador (que indica o quanto a informação sensorial é processada antes de se chegar ao atuador) e um aumento na velocidade de resposta da arquitetura, ao passo que a capacidade de predição dos resultados a serem obtidos e a dependência que a arquitetura possui acerca da confiabilidade do modelo do mundo (ambiente de operação do robô) diminuem.

Os primeiros robôs móveis desenvolvidos usavam arquiteturas de navegação deliberativas. Estas são oriundas da escola de Inteligência Artificial (IA) e, de alguma forma, são inspiradas na maneira como os seres humanos raciocinam. Portanto, estas arquiteturas fazem uso intensivo de representações simbólicas do conhecimento, sobretudo no que tange ao modelo do mundo, ou seja, o ambiente de operação do robô móvel. Como características principais, as arquiteturas deliberativas são relativamente lentas, devido principalmente à sua estrutura de fluxo de informação sequencial e ao intenso uso de representação simbólica de conhecimento. Isso possibilita o planejamento prévio das ações do robô, permitindo a execução de tarefas de forma ótima, porém bastante sensível a alterações no modelo do ambiente utilizado para realizar o planejamento, o que caracteriza este tipo de arquitetura como pouco robusta.

No outro extremo do espectro estão as arquiteturas reativas. Em seu caso extremo, também conhecido como reflexivo, não existe qualquer tipo de representação simbólica de conhecimento e, portanto, nenhum tipo de modelo de mundo. O acoplamento sensor-atuador é muito forte, o que em parte é o reflexo de um fluxo de

informação paralelo, em contraposição ao fluxo sequencial das arquiteturas deliberativas. Tais características resultam em arquiteturas adequadas para aplicações em ambientes dinâmicos, onde a resposta do sistema em tempo real é essencial para evitar colisões e garantir a realização da tarefa especificada para o robô, privilegiando assim a robustez do sistema. Em contrapartida, é extremamente difícil prever os resultados que podem ser obtidos aplicando arquiteturas deste tipo, o que dificulta a implementação de um MSA otimizado.

Entre os dois extremos do espectro estão as arquiteturas denominadas híbridas, que não são nem puramente deliberativas, nem puramente reativas. Dentre elas, pode haver uma maior predominância da parte deliberativa ou reativa. Na verdade, um espectro contínuo como apresentado na figura 2.2 tem como objetivo principal ilustrar justamente essa transição suave entre as arquiteturas deliberativas e reativas; mesmo porque, é muito difícil dizer a partir de que ponto uma arquitetura deixa de ser deliberativa e passa a ser híbrida, ou deixa de ser híbrida e passa a ser reativa.

2.1.1 Arquiteturas deliberativas

As arquiteturas de navegação que utilizam metodologia deliberativa consistem em um conjunto de blocos funcionais em sequência em que a informação do ambiente, proveniente dos sensores, é normalmente utilizada para a construção de um modelo do ambiente de navegação do robô. Este modelo deve incluir informações a respeito das dimensões, formas, posições e orientações de todos os objetos de interesse presentes na área de operação do robô móvel.

Muitas vezes, o modelo do ambiente é conhecido *a priori*, seja por intermédio de uma planta baixa do ambiente de operação, ou pelo uso de informações obtidas em tarefas realizadas anteriormente e armazenadas na memória do robô [23, 24]. Nesses casos, este mapa pode ser utilizado para planejar a tarefa de forma global, ficando o sensor responsável pela localização do robô no ambiente, pelas tarefas reativas do sistema, como por exemplo evitar obstáculos, e pela atualização do modelo *a priori* do ambiente, incorporando ao mapa inicial informações novas como paredes e posicionamento dos obstáculos encontrados durante a navegação.

Utilizando o modelo do ambiente, é possível definir, por intermédio de um planejamento de trajetória, as atitudes do robô de forma ótima, as quais deverão ser executadas pelos atuadores. Um exemplo da estrutura de uma arquitetura de navegação deliberativa está representado na figura 2.3. É importante ressaltar que, não necessariamente, todos os blocos funcionais mostrados na figura devem estar presentes em uma arquitetura de navegação deliberativa. Esse sistema de navegação é indicado para tarefas de alto nível, executadas em ambientes estáticos, como planejamento global ou cronograma de atividades. A sua utilização para tarefas em

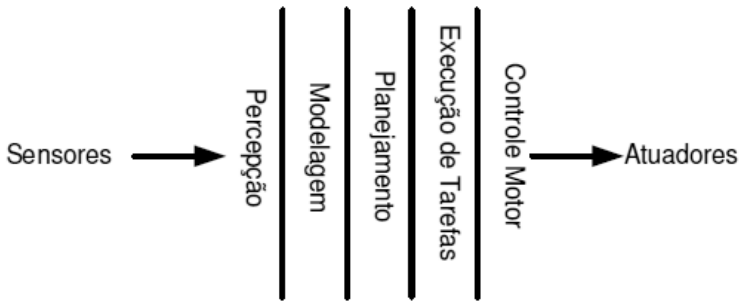


Figura 2.3: Estrutura de uma arquitetura de navegação deliberativa.

ambientes dinâmicos não é indicada, devido à sua natureza sequencial, o que exige um tempo de execução elevado, impossibilitando, assim, a obtenção de respostas satisfatórias às alterações imprevisíveis no ambiente. Exemplos desses sistemas de navegação podem ser encontrados em [25–30]. As principais vantagens das arquiteturas de navegação deliberativas são:

1. "Simplicidade" na interação entre as etapas do processo (*sense-plan-act*);
2. Execução sequencial em cadeia fechada;
3. Possibilita um comportamento ótimo, ou racional;
4. Utilização eficiente dos mecanismos disponíveis.

Em contrapartida, esse tipo de arquitetura apresenta as seguintes desvantagens:

1. É inapropriada para sistemas que necessitam de resposta rápida (ambientes dinâmicos, por exemplo);
2. Necessita de informações confiáveis e estáticas para a construção correta do modelo do ambiente.

2.1.2 Arquiteturas reativas

A principal característica das arquiteturas reativas é o forte acoplamento entre sensoriamento e atuação, não sendo utilizados modelos do mundo. Seus defensores argumentam que as arquiteturas reativas, ao invés de operar com base em representações abstratas da realidade (modelo do ambiente), operam com base na própria realidade. Com isso, evita-se o alto custo computacional envolvido na elaboração e manutenção de modelos confiáveis do ambiente de operação do robô. Outra característica comum é que normalmente estas arquiteturas consistem em simples métodos

baseados em regras, de baixo custo computacional. Dessa forma, as arquiteturas reativas tendem a ser rápidas e, portanto, adequadas para aplicações onde o ambiente de operação é altamente dinâmico e/ou não estruturado.

Ao contrário da metodologia deliberativa, essa metodologia é indicada para ambientes onde a velocidade de resposta é crucial para o funcionamento correto do sistema, mas não é indicada para tarefas de alto nível que exijam otimalidade. Exemplos de arquiteturas de navegação reativa podem ser encontrados em [2, 31–35].

As principais vantagens das arquiteturas reativas são:

1. Velocidade (resposta rápida a estímulos externos);
2. Forte acoplamento entre percepção e ação;
3. Independência em relação ao modelo do ambiente de atuação.

Em contrapartida esse tipo de arquitetura apresenta as seguintes desvantagens:

1. Ausência de memória;
2. Dificuldade na realização de tarefas globalmente complexas;
3. Inapropriada para tarefas de alto nível que exijam otimalidade em algum aspecto.

2.1.3 Arquiteturas híbridas

A partir da descrição dos MSA deliberativos e reativos, é possível notar que o primeiro privilegia a eficiência em detrimento da velocidade, enquanto o segundo privilegia a velocidade em detrimento da eficiência. Nesse contexto, os sistemas híbridos surgem como arquiteturas de navegação que tentam unir as melhores características das duas metodologias, possibilitando uma navegação mais robusta e eficiente.

A parte reativa é, geralmente, responsável pela navegação segura, possibilitando que o robô execute tarefas em tempo real ou em situações de emergência, enquanto a parte deliberativa do sistema é responsável pelo cumprimento da tarefa como um todo, guiando o sistema reativo para atingir certos objetivos específicos, garantindo sempre que possível a utilização eficiente dos recursos disponíveis no robô. No entanto, o critério de distribuição das funcionalidades requeridas para uma arquitetura híbrida entre seus componentes deliberativos e reativos, e a forma como é feita a interface entre tais componentes, tão distintos em sua essência, ainda são aspectos que não foram completamente entendidos, e portanto representam áreas abertas para a realização de novas pesquisas. Exemplos de arquiteturas híbridas podem ser encontrados em [36–42].

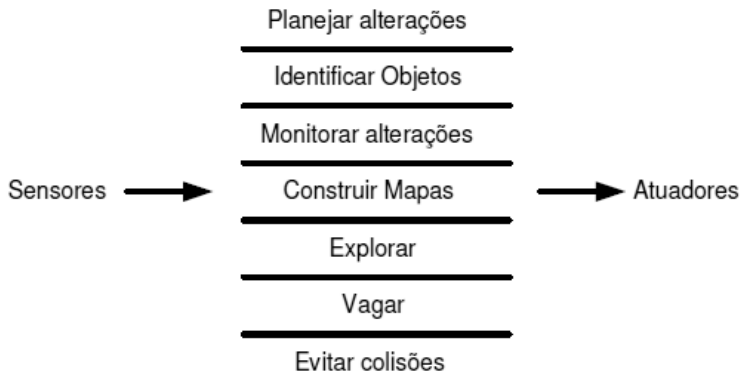


Figura 2.4: Arquiteturas de navegação baseadas em comportamentos.

2.2 Arquiteturas baseadas em comportamentos

Muitas vezes, robôs móveis são destinados a navegar em ambientes complexos e dinâmicos com o objetivo de realizar uma tarefa igualmente complexa e específica. Em situações como esta, o desenvolvimento de um único sistema de controle capaz de fazer com que o robô alcance seu objetivo final nem sempre é possível. Neste sentido, diversos trabalhos foram desenvolvidos com o intuito de decompor uma tarefa complexa (tarefa global) em diversas tarefas mais simples (sub-tarefas), possibilitando o desenvolvimento de sistemas de controle independentes, reativos ou deliberativos. Estes sistemas são capazes de satisfazer as especificações definidas para cada uma das sub-tarefas, caracterizando, assim, uma navegação descentralizada [2].

Os sistemas de navegação que visam a realização de uma sub-tarefa, em detrimento da tarefa global são chamados de comportamentos [2]. Nesse tipo de arquitetura, cada comportamento é executado em paralelo e os sensores e atuadores interagem diretamente com todas as camadas. Isso permite que novas camadas de comportamento possam ser adicionadas ao sistema, sem que haja a necessidade de reconfigurá-lo por completo. Dessa forma comportamentos inteligentes e abrangentes podem ser construídos através de uma composição de comportamentos mais simples. Um exemplo da execução paralela, que caracteriza este tipo de arquitetura é apresentado na figura 2.4.

Frequentemente, arquiteturas baseadas em comportamento e arquiteturas reativas são consideradas como sendo a mesma coisa. No entanto, isso não é correto. Por exemplo, a arquitetura de navegação denominada Desvio Tangencial de Obstáculos [43] é reativa, mas não pode ser considerada baseada em comportamentos, pois uma premissa básica para estas é que devem coexistir vários módulos, ou comportamentos, sendo executados em paralelo, o que não ocorre no Desvio Tangencial de Obstáculos, em que existe um único módulo. Do contrário, também existem arquiteturas baseadas em comportamentos que possuem componentes de planejamento,

ou seja, deliberativos, o que obviamente exclui a possibilidade de que este tipo de arquitetura seja puramente reativa. Considerar arquiteturas reativas e baseadas em comportamento como sinônimos é um erro decorrente do fato de que, em ambos os casos, busca-se um forte acoplamento sensor-atuador. Também é fato que muitas arquiteturas baseadas em comportamento são de fato reativas. No entanto, nada impede que uma arquitetura baseada em comportamentos seja constituída por um ou mais componentes deliberativos, associados ou não a componentes reativos.

Junto com os benefícios trazidos pelas arquiteturas de navegação baseadas em comportamentos, surge uma questão que é tema de trabalho de inúmeros pesquisadores em todo o mundo [2]:

> *"Como compor, ou coordenar, da melhor forma possível, os comportamentos individuais para atingir o objetivo global de um sistema de navegação?"*

Infelizmente, não existe atualmente uma resposta universal para esta pergunta. No entanto, diversas técnicas foram desenvolvidas e metodologias foram propostas para realizar a coordenação de comportamentos. Um ampla revisão bibliográfica sobre o tema pode ser encontrada em [3] e [2].

Uma arquitetura de navegação baseada em comportamentos pode ser classificada, qualitativamente, de acordo com a forma de selecionar os comportamentos, que devem agir em cada situação (figura 2.5), podendo esta ser competitiva (métodos de arbitração) ou cooperativa (métodos de fusão de comandos) [3]. Nos métodos de arbitração, um único comportamento por vez é responsável pela atuação em todo o sistema. Os métodos baseados em arbitração diferem entre si pela forma como o controlador é selecionado. As três principais formas de arbitração são: (*i*) os sistemas baseados em prioridades [31]; (*ii*) *winner takes all* [22] e; (*iii*) sistemas baseados em estados [6]. Nos métodos de fusão de comandos, um conjunto de controladores comanda todo o sistema ao mesmo tempo. As formas mais comuns de fusão de sinais de controle são por votação [36], por superposição [34, 41], por lógica fuzzy [39] e por múltiplos objetivos [44]. Métodos mais recentes levam em consideração informações estatísticas dos sinais de controle para realizar a fusão, como é o caso dos métodos baseados em filtro de Kalman [45] e em filtro de informação [33].

A arquitetura de navegação desenvolvida neste trabalho é uma arquitetura prioritariamente deliberativa, que utiliza a teoria de sistemas a eventos discretos e controle supervisório [4, 12, 13], para realizar a composição dos comportamentos modelados e o planejamento da trajetória. No entanto, para melhor detalhar o MSA proposto, é necessário apresentar os fundamentos de sistemas a eventos discretos que foram utilizados no desenvolvimento da arquitetura de navegação proposta.

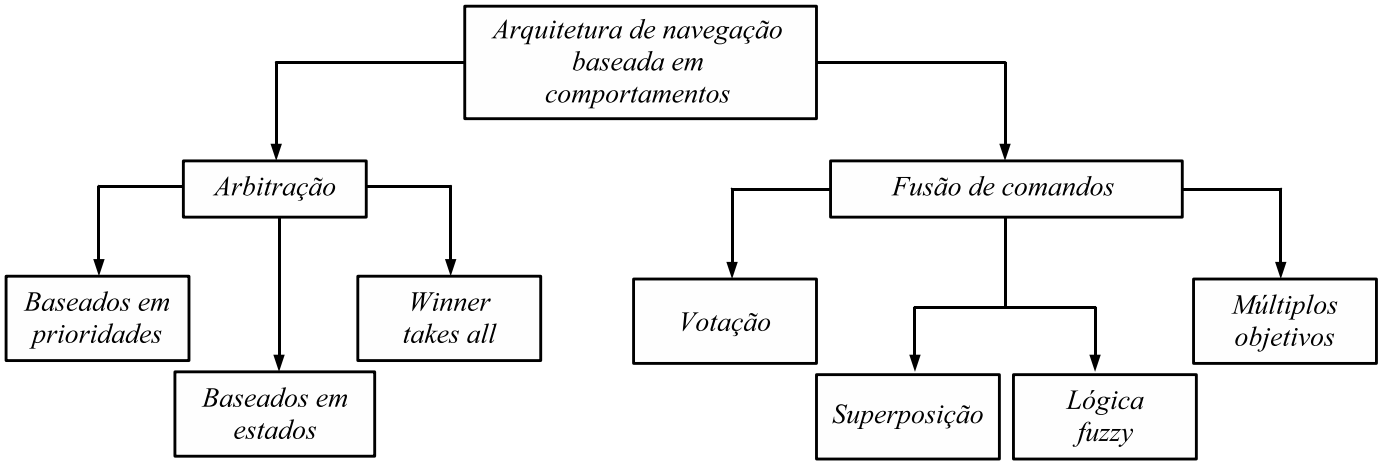


Figura 2.5: Classificação das arquiteturas de navegação baseadas em comportamentos, segundo Pirjanian [3].

Capítulo 3

Fundamentos de sistemas a eventos discretos

Historicamente, cientistas e pesquisadores têm concentrado seus esforços em estudar e compreender fenômenos naturais que podem ser, na maioria das vezes, modelados e analisados através de teorias já consolidadas como a lei da gravidade, mecânica clássica e não-clássica, físico-química, entre outras. Assim sendo, ao tentar solucionar um problema físico, é comum deparar-se com variáveis como posição, orientação, velocidade e aceleração. Essas variáveis evoluem continuamente no tempo, isto é, podem assumir qualquer valor real à medida que o tempo passa. Baseado nessa idéia, uma vasta base de ferramentas e técnicas matemáticas foram desenvolvidas para modelar, analisar e controlar os diferentes sistemas encontrados no mundo real. Dentro desta base matemática, as equações diferenciais representam a ferramenta mais completa para analisar e controlar sistemas a variáveis contínuas.

Com o passar do tempo, a complexidade dos processos a serem analisados pelo homem cresceu rapidamente, e com ela, cresceu também a necessidade e a dependência humana em relação aos mecanismos capazes de processar essa informação. Dessa forma, há uma demanda crescente por computadores cada vez mais potentes, estabelecendo-se assim uma relação de dependência entre o homem e a máquina. No entanto, apesar de serem capazes de modelar e simular sistemas de complexidade elevada, tarefa que seria praticamente impossível sem o auxílio destes, os computadores possuem uma limitação que pode muitas vezes interferir no resultado de um problema: a incapacidade de representar grandezas contínuas. Essa limitação impulsiona o desenvolvimento de novas áreas, onde as variáveis e os modos de operação são considerados de maneira discreta, desde a concepção do modelo até a sua implementação.

Dependendo do processo a ser estudado, é necessário realizar uma análise quantitativa do sistema, para determinar as variáveis que mais influenciam o seu comportamento, ou mesmo para possibilitar o projeto de um sistema de controle, por

exemplo. Assim sendo, é necessário encontrar um modelo que seja capaz de representar de forma concisa o comportamento de um sistema e suas reações a diferentes estímulos externos. Intuitivamente, é possível pensar em um modelo como sendo um dispositivo que reproduz o comportamento do sistema a ser analisado.

As variáveis necessárias em um modelo para reproduzir adequadamente o comportamento de um sistema são as chamadas variáveis de estados. O conjunto de todas as variáveis de estado de um sistema é denominado estado de um sistema. A depender da natureza do sistema modelado o estado de um sistema pode ser contínuo (quando as variáveis de estado podem assumir qualquer valor real) ou discreto (quando as variáveis de estado podem assumir apenas valores específicos). Já a dinâmica do sistema pode ser modelada de duas formas: determinada pelo tempo (quando a mudança no estado do sistema está associada a uma mudança no tempo), ou determinada por eventos (quando a mudança no estado do sistema está associada à ocorrência de um fato específico que não, necessariamente, dependa do tempo).

Este capítulo está organizado da seguinte forma. Na seção 3.1 é apresentada a modelagem de sistemas a eventos discretos. Na seção 3.2 é introduzido o conceito de linguagem, ilustrando as operações que podem ser realizadas com ela. Na seção 3.3 é apresentada a teoria de autômatos, destacando a sua capacidade de representar graficamente uma linguagem. Finalmente, na seção 3.4 são apresentadas as operações que podem ser realizadas com um, ou mais de um, autômato.

3.1 Modelagem de um sistema

Para desenvolver a modelagem de um processo, é necessário primeiramente definir as variáveis de interesse do sistema. Analisando estas variáveis em um período de tempo $[t_0,t_f]$, é possível coletar informações que permitirão ao projetista desenvolver um modelo adequado a este sistema. Essas variáveis podem ser inicialmente separadas em dois grupos: variáveis de entrada e variáveis de saída, denotadas respectivamente por $u(t)$ e $y(t)$. Variáveis de entrada são as variáveis passíveis de alteração por um agente externo, ou seja, seu valor pode ser alterado ao longo do tempo independentemente da reação do sistema a esta alteração. Já as variáveis de saída são aquelas cujo valor é alterado pelo comportamento do sistema em resposta ao estímulo das variáveis de entrada. Para fins de simplificação de notação, as variáveis de entrada são representadas na forma de um vetor coluna $\mathbf{u}(t)$ e as variáveis de saída na forma de outro vetor coluna $\mathbf{y}(t)$, da seguinte forma:

$$\mathbf{u}(t) = [u_1(t), u_2(t), \ldots, u_p(t)]^T \tag{3.1}$$

$$\mathbf{y}(t) = [y_1(t), y_2(t), \ldots, y_m(t)]^T. \tag{3.2}$$

É possível ainda classificar as variáveis de um sistema em um terceiro grupo, não exclusivo em relação aos outros grupos: as variáveis de estado, denotadas por $x(t)$. O conjunto formado pelas variáveis de estado de um sistema, no instante t, é chamado de estado do sistema, $\mathbf{x}(t)$, que, assim como $\mathbf{u}(t)$ e $\mathbf{y}(t)$, pode ser representado na forma de um vetor, isto é:

$$\mathbf{x}(t) = [x_1(t), x_2(t), \ldots, x_n(t)]^T. \tag{3.3}$$

O estado de um sistema no instante t_0, é definido em [4] como sendo a informação necessária em t_0 para que a saída $y(t)$ seja unicamente determinada, para todo instante $t \geq t_0$, a partir dessa mesma informação e da entrada $u(t)$ nesse mesmo intervalo de tempo.

Definição 3.1 *(Espaço de estados) O espaço de estados de um sistema, usualmente denotado por X, é o conjunto de todos os possíveis valores que os elementos do estado de um sistema, $\mathbf{x}(t)$, podem assumir para um dado valor de t.*

Para completar o modelo de um sistema, é necessário, agora, formular a relação que existe entre as diferentes variáveis de interesse. A forma mais usada para definir essa relação é através do modelo em espaço de estados. Essa metodologia consiste em determinar a relação matemática apropriada envolvendo a entrada $u(t)$, a saída $y(t)$ e o estado $x(t)$ do sistema. Essa relação é chamada de dinâmica do sistema.

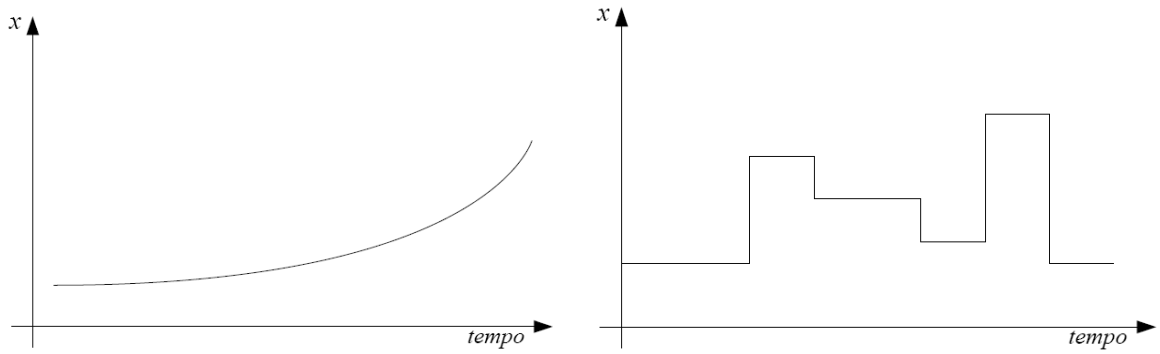
A dinâmica de um sistema é definida por um grupo de equações de estado, necessárias para especificar o valor do estado $x(t)$, para qualquer valor de $t \geq t_0$, dado $x(t_0)$ e a função de entrada $u(t)$ no mesmo intervalo de tempo. As equações de estado podem ser de diferentes formas, no entanto, a forma mais utilizada para definir essas relações é utilizando equações diferenciais. Somente será possível dizer que um modelo em espaço de estados para um sistema foi obtido, quando forem completamente especificadas as seguintes equações:

$$\dot{\mathbf{x}}(t) = f[\mathbf{x}(t), \mathbf{u}(t), t], \qquad \mathbf{x}(t_0) = \mathbf{x}_0 \tag{3.4}$$

$$\mathbf{y}(t) = g[\mathbf{x}(t), \mathbf{u}(t), t], \tag{3.5}$$

em que a primeira compreende as equações de estado com as condições iniciais especificadas e a segunda as equações de saída, que determinam as variáveis de saída do sistema em função do tempo e das variáveis de estado e de estrada.

O uso de equações diferenciais na modelagem em espaço de estados limita a aplicação dessa metodologia aos sistemas cujo espaço de estados é contínuo, ou seja, o estado do sistema pode assumir qualquer valor, normalmente real, dentro de um intervalo. Nesse caso, a trajetória do estado ao longo do tempo pode ser representada por uma função contínua por partes, ou por uma única função contínua,



(a) (b)

Figura 3.1: (a) Trajetória de estado para espaço de estado contínuo; (b) Trajetória de estado para espaço de estado discreto.

como ilustrado na figura 3.1(a). No entanto, a depender do problema a ser modelado o estado precisa assumir valores discretos quantitativos, como números inteiros por exemplo, ou mesmo valores qualitativos, como {LIGADO, DESLIGADO} ou {ALTO, MÉDIO, BAIXO}. Quando o estado de um sistema pode assumir apenas um número limitado de valores ao longo do tempo, esse sistema é chamado de *sistema de estados discretos*. Nesse caso a trajetória do estado do sistema ao longo do tempo é normalmente representada utilizando uma função constante por partes (figura 3.1(b)), visto que ao estado somente será permitido mudar de um valor para outro em instantes discretos de tempo.

A dinâmica de um sistema de estados discretos é bem mais simples de visualizar do que a de um sistema de estados contínuos. Isso se deve ao fato de que o mecanismo de transição de estados, em um sistema a estados discretos, é normalmente baseado em expressões lógicas simples, do tipo "se acontecer alguma coisa $\boldsymbol{e}$ e o estado atual do sistema for $\mathbf{x} = \mathbf{q_1}$, então o proximo valor do estado será $\mathbf{x} = \mathbf{q_2}$". Entretanto, o ferramental necessário para expressar formalmente estes sistemas e encontrar as soluções necessárias para um dado problema, pode vir a ser consideravelmente mais complexo.

Na maior parte das vezes supõe-se que o tempo seja uma variável contínua. No entanto, às vezes é preciso determinar os valores das variáveis de interesse apenas em instantes discretos de tempo. A esses sistemas dá-se o nome de *sistemas de tempo discreto*. Muitos motivos apontam para a utilização desse tipo de mecanismo para modelar os processos no nosso dia-a-dia, como a popularização dos computadores digitais, a facilidade na obtenção de soluções numéricas para equações de alta complexidade, a velocidade no controle de processos obtida através de técnicas de controle digital, entre outros. Vale ressaltar que a discretização do tempo não implica a discretização do espaço de estados de um sistema, tampouco o contrário é verdadeiro.

Nos sistemas em que o espaço de estados é discreto, pode-se observar que as transições de estado ocorrem instantaneamente. Aos instantes discretos de tempo em que as transições ocorrem é associado um evento. Um evento pode ser identificado como a ocorrência de uma ação específica como por exemplo apertar um botão, o motor de uma máquina parar, a luz do semáforo ficar verde ou uma variável atingir um dado *setpoint*. Quando em um sistema a transição de estados estiver associada à variável independente tempo, dizer-se-á que este sistema é *dirigido pelo tempo*. Quando a transição de estados estiver associada à ocorrência de um evento, este sistema será chamado de sistema *dirigido por eventos*.

Nesse ponto é possível apresentar uma definição para sistemas a eventos discretos [4, 13]:

Definição 3.2 *(Sistema a eventos discretos) Um sistema a eventos discretos (SED) é um sistema de estado discreto e dirigido por eventos, isto é, a evolução dos seus estados depende exclusivamente da ocorrência de eventos discretos e, em geral, assíncronos em relação à escala de tempo.*

Para compreender melhor o comportamento dos sistemas a eventos discretos, foram desenvolvidas formas apropriadas de modelagem e representação destes, fornecendo assim ao projetista as ferramentas necessárias para melhor analisar e controlar processos dessa natureza. Nas seções seguintes serão abordadas as ferramentas e os formalismos de SED utilizados no desenvolvimento deste trabalho.

3.2 Linguagem de um sistema

Ao analisar a evolução de estados de um sistema a evento discretos, existem duas preocupações principais: qual a sequência de estados visitados do sistema e quais eventos estão associados a esta sequência. Qualquer sistema a eventos discretos possui um conjunto de eventos $E = \{e_1, e_2, \ldots, e_n\}$ associados a ele. A este conjunto dá-se o nome de *alfabeto*. Em todas as observações apresentadas neste trabalho supõe-se que E é um conjunto finito. Supondo que um dado comportamento de um SED pode ser descrito por uma sequência de eventos, *palavra*, do tipo $e_1 e_2 \ldots e_n$, então, o conjunto de todos os possíveis comportamentos de um sistema pode ser definido por um conjunto de palavras, denominado *linguagem*.

Sobre um mesmo conjunto de eventos E é possível definir diferentes linguagens. Por exemplo, seja um conjunto de eventos definidos por $E = \{a, b, c\}$. Então é possível definir uma linguagem L_1 contendo todas as possíveis palavras de comprimento três que comecem com o evento a. Essa linguagem possui nove palavras, qual seja:

$$L_1 = \{aaa, aab, aac, aba, abb, abc, aca, acb, acc\}. \tag{3.6}$$

Pode-se ainda, definir uma outra linguagem L_2 que contém todas as palavras, de comprimento finito, que comecem com o evento a. Esta por sua vez possui infinitas palavras, isto é:

$$L_2 = \{a, aa, ab, ac, aaa, aab, aac, aba, abb, abc, aca, acb, acc, aaaa, aaab, \ldots\}. \quad (3.7)$$

A seguir serão apresentadas algumas definições necessárias para o entendimento e manipulação de linguagens.

Definição 3.3 *[4]:*

1. *O comprimento de uma palavra s é o número de eventos contidos nessa palavra. Esse comprimento é representado pela notação $|s|$.*

2. *ε indica a palavra vazia e tem comprimento igual a zero, isto é, $|s| = 0$. Representa uma sequência que não possui eventos.*

3. *Concatenação: É a operação chave utilizada para criar uma palavra. Por exemplo a palavra abb é a concatenação da palavra ab com o evento (ou palavra de comprimento um) b.*

4. *ε é a operação identidade da concatenação, ou seja:*

$$s\varepsilon = \varepsilon s = s, \qquad \forall s. \quad (3.8)$$

5. *(Fecho de Kleene) O Fecho de Kleene de um conjunto de eventos E, é o conjunto de todas as palavras de comprimento finito geradas a partir dos elementos de E, incluindo a palavra vazia ε. A notação E^* representa a operação Fecho de Kleene sobre E. Por exemplo, se $E = \{a, b, c\}$, então:*

$$E^* = \{\varepsilon, a, b, c, aa, ab, ac, bb, ba, bc, cc, ca, cb, aaa, \ldots\}. \quad (3.9)$$

   Assim sendo, a maior linguagem que pode ser definida sobre um conjunto de eventos E é E^* e qualquer outra linguagem é um subconjunto de E^*.

6. *dada a palavra $s = tuv$, com $s, t, u, v \in E^*$, então:*

   - *t e tu são exemplos de prefixos de s;*
   - *u é uma sub-palavra de s;*
   - *v e uv são exemplos de sufixos de s.*

3.2.1 Operações com linguagens

As operações usuais da teoria de conjuntos, tais como união, intersecção, diferença e complemento com respeito a E^* também se aplicam a linguagens, dado que uma linguagem é também, por definição, um conjunto cujos elementos são palavras. Além dessas operações, é preciso definir, pelo menos, outras três operações de extrema importância no estudo de sistemas a eventos discretos. São elas: concatenação, fecho do prefixo e fecho de Kleene (para linguagens). Essas definições são apresentadas na sequência.

Definição 3.4 *[4]:*

1. *(Concatenação) Seja $L_a, L_b \subseteq E^*$. Então uma palavra s pertence à concatenação das linguagens L_a e L_b, se ela puder ser escrita como a concatenação de uma palavra pertencente a L_a com outra pertencente a L_b, ou seja:*

$$L_a L_b := \{s \in E^* : (\exists s_a \in L_a)(\exists s_b \in L_b)[s = s_a s_b]\}. \tag{3.10}$$

2. *(Fecho do Prefixo) Seja $L \subseteq E^*$. Então o fecho em prefixo de L, denotado por $\overline{L}$, é a linguagem formada por todos os prefixos de todas as palavras pertencentes a L. Em geral, $L \subseteq \overline{L}$. Assim:*

$$\overline{L} := \{s \in E^* : (\exists t \in E^*) \quad [st \in L]\}. \tag{3.11}$$

   Obs.: L é dita de prefixo fechado quando $L = \overline{L}$, ou seja, se qualquer prefixo de qualquer palavra pertencente a L, também pertencer a L.

3. *(Fecho de Kleene) Seja $L \subseteq E^*$. Então o fecho de Kleene de L, denotado por L^*, é o conjunto formado por todas as possíveis concatenações entre palavras pertencentes a L. Essa operação pode ser expressa da seguinte forma:*

$$L^* := \{\varepsilon\} \quad \cup \quad L \quad \cup \quad LL \quad \cup \quad LLL \quad \cup \quad \ldots. \tag{3.12}$$

   Assim como no caso do fecho de Kleene para um conjunto de eventos E, o fecho de Kleene para linguagens deve incluir o elemento ε. Vale a pena ressaltar que o fecho de Kleene é uma operação idempotente, isto é $(L^*)^* = L^*$.

Considere como exemplo, o conjunto de eventos $E = \{a, b, c\}$ e as linguagens definidas sobre este conjunto $L_1 = \{\varepsilon, a, abb\}$ e $L_2 = \{c\}$. Dessa forma, é possível

ilustrar algumas das operações definidas acima, como segue:

$$\begin{aligned} L_1 L_2 &= \{c, ac, abbc\} \\ \overline{L_1} &= \{\varepsilon, a, ab, abb\} \\ \overline{L_2} &= \{\varepsilon, c\} \\ L_1 \overline{L_2} &= \{\varepsilon, a, abb, g, ag, abbg\} \\ L_2^* &= \{\varepsilon, c, cc, ccc, cccc, \ldots\} \\ L_1^* &= \{\varepsilon, a, abb, aa, aabb, abba, abbabb, \ldots\} \\ E^* &= \{\varepsilon, a, b, c, aa, ab, ac, ba, bb, bc, ca, cb, cc, aaa, \ldots\}. \end{aligned} \tag{3.13}$$

Note que nem L_1, nem L_2 são de prefixos fechados, uma vez que $ab \notin L_1$ e $\varepsilon \notin L_2$. Note também que tanto L_1^*, quanto L_2^* são subconjuntos de E^*. O mesmo vale para qualquer linguagem L_i definida sobre o conjunto de eventos E, em que a relação $L_i^* \subseteq E^*$ sempre é verdadeira [4].

3.2.2 Representação de uma linguagem

Como descrito ao longo da seção 3.2, uma linguagem pode ser vista como um método formal de descrever o comportamento de um sistema a eventos discretos. Ela especifica todas as sequências de eventos admissíveis que um dado SED pode executar, ou seja, todos os possíveis comportamentos do sistema. Ao observar as equações (3.6) e (3.7), é possível concluir que a linguagem L_1 é facilmente representada por uma simples enumeração das palavras contidas nela, pois esta possui um número finito de elementos (nove). Já para a linguagem L_2, não é possível realizar o mesmo procedimento, podendo esta ser definida apenas descritivamente.

Essa dificuldade na representação de uma linguagem demonstra a necessidade de um formalismo capaz de fornecer o ferramental necessário, não só para expressá-las, mas também para possibilitar a realização de operações com linguagens de forma simples, permitindo assim a construção, análise e manipulação de sistemas a eventos discretos representados por linguagens extremamente complexas.

Na seção 3.3 é apresentada a teoria de Autômatos como uma ferramenta capaz de representar e possibilitar a manipulação de uma dada linguagem, permitindo assim resolver os problemas relacionados ao comportamento de uma linguagem associada a um dado SED.

3.3 Autômatos

Um autômato, também conhecido como máquina de estados, é um dispositivo capaz de representar uma linguagem de acordo com regras bem definidas. A forma mais

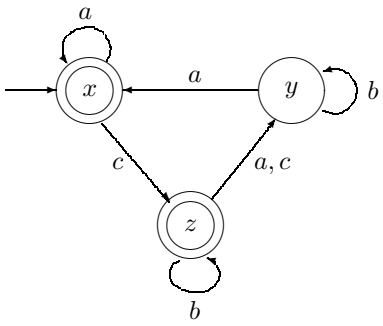


Figura 3.2: Diagrama de transição de estados do autômato G_1.

simples de apresentação de um autômato é a sua representação gráfica, através de um diagrama de transição de estados, como o ilustrado na figura 3.2. Nessa representação, cada nó está associado a um estado do sistema e as transições estão associadas aos eventos que promovem a mudança de um estado para outro. Essa representação possibilita uma descrição visual da dinâmica do sistema, capturada pelo autômato.

Um autômato G é uma sêxtupla $G = (X, E, f, \Gamma, x_0, X_m)$ em que [4]:

1. X é o espaço de estados do sistema;

2. E é o conjunto finito de eventos associados a G;

3. f é a função de transição de estados definida como:

$$f : X \times E \to X \tag{3.14}$$

$$(x, e) \longmapsto y = f(x, e), \tag{3.15}$$

   que significa que existe uma transição associada a um evento e que, acontecendo no estado x, leva o sistema para o estado y;

4. Γ é a função que determina os eventos ativos em um estado e é definida em $\Gamma : X \to 2^E$. Ou seja, $\Gamma(x)$ é o conjunto de todos os eventos $e \in E$ para os quais $f(x, e)$ é definida, $\forall x \in X$;

5. x_0 é o estado inicial ou estado de partida do sistema. No diagrama de transição de estados, o estado inicial é aquele que possui uma "seta livre" apontada para ele;

6. X_m é um subconjunto de X, denominado de conjunto dos estados marcados. Um estado pode ser marcado por diversos motivos como, por exemplo, para ressaltar um estado que indica o término de uma tarefa, ou para indicar que o sistema está em uma situação crítica. A decisão de quais estados devem ou

não ser marcados é uma atribuição do projetista. No diagrama de transição de estados, os estados marcados são aqueles identificados com um duplo círculo.

Utilizando a definição de autômatos apresentada, é possível analisar o diagrama de transição de estados apresentado na figura 3.2 e definir explicitamente, para este exemplo, cada uma das sete componentes de G_1. O conjunto de eventos associados a G_1 é dado por $E = \{a, b, c\}$, o espaço de estados do sistema é $X = \{x, y, z\}$, o estado inicial é $x_0 = \{x\}$ e o conjunto de estados marcados é $X_m = \{x, z\}$. A informação da dinâmica do sistema (contida em f e Γ), é representada no diagrama de transições de estados pelos arcos direcionais (transições). Os valores de $f(x_i, e)$, $\forall x_i \in X$ e $\forall e \in E$, são $f(x, a) = x$, $f(x, c) = z$, $f(y, a) = x$, $f(y, b) = y$, $f(z, b) = z$, $f(z, a) = y$ e $f(z, c) = y$. Os valores de $\Gamma(x_i)$, $\forall x_i \in X$, são $\Gamma(x) = \{a, c\}$, $\Gamma(y) = \{a, b\}$ e $\Gamma(z) = E = \{a, b, c\}$.

Um autômato G funciona sempre de maneira sequencial, iniciando seu estado a partir de x_0. O estado do sistema permanece o mesmo até que ocorra um evento $e \in \Gamma(x_0) \subseteq E$. Quando da ocorrência deste evento, acontece uma transição de estado $f(x_0, e) = x_i \in X$ (podendo ser $x_i = x_0$), que levará o sistema para o estado x_i. Esse processo acontece continuamente enquanto o sistema estiver em operação.

O tipo de autômato que será utilizado neste trabalho é o *autômato determinístico* e de *espaço de estados finitos*. Determinístico porque a ocorrência de um dado evento e em um determinado estado x_1, implica uma transição de estado do tipo $f(x_1, e) = x_2$, onde x_2 é unicamente determinado, podendo este assumir qualquer valor dentro do espaço de estados do sistema, inclusive x_1. E de estados finitos por que seu espaço de estados é um conjunto com um número finito de elementos. Na sequência deste trabalho os termos *autômato* e *autômato determinístico de estados finitos* serão utilizados indiscriminadamente.

3.3.1 Autômato como representação de uma linguagem

Uma vez explicada a definição de um autômato e suas características, é preciso estabelecer uma ligação entre um autômato e a linguagem a qual se deseja representar. Essa relação é facilmente construída observando-se o diagrama de transição de estados de um autômato G qualquer.

O conjunto de todas as possíveis trajetórias de estado (definidas por uma sequência de eventos), partindo do estado inicial x_0, corresponde à *linguagem gerada* do sistema representado pelo autômato G. O conjunto de todas as possíveis sequências de eventos que levam o sistema do estado inicial x_0 até um estado marcado $x \in X_m$, corresponde à *linguagem marcada* do sistema representado por G. Quando uma linguagem pode ser marcada por um autômato de estados finitos, ela é denominada *linguagem regular*.

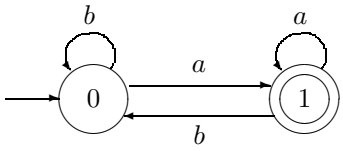


Figura 3.3: Autômato G_2 que marca a linguagem L.

Formalmente, a *linguagem gerada* por $G = (X, E, f, \Gamma, x_0, X_m)$ é dada por

$$\mathcal{L}(G) := \{s \in E^* : f(x_0, s) \;\; \text{é} \;\; definida\}, \tag{3.16}$$

e a *linguagem marcada* por G é definida como sendo

$$\mathcal{L}_m(G) := \{s \in \mathcal{L}(G) : f(x_0, s) \in X_m\}. \tag{3.17}$$

Essas definições utilizam uma extensão da função de transição de estados f definida anteriormente na equação (3.14). Essa extensão nada mais é do que uma generalização de f, para os casos em que o segundo argumento não tem comprimento igual a um. Ao usar essa generalização não é necessária uma nova notação para f, pois ambas definições são consistentes para o caso de um único evento. Assim f pode ser definida, sem perda de generalidade, em $f : X \times E^* \to X$ da mesma forma.

Um autômato G pode ser resumido como sendo a representação de duas linguagens: $\mathcal{L}(G)$ e $\mathcal{L}_m(G)$. Por exemplo, considere a linguagem:

$$L = \{a, aa, ba, aaa, aba, baa, bba, \ldots\}, \tag{3.18}$$

que é formada por todas as palavras, de comprimento não nulo, formadas por a e b ($\{a, b\}^* - \varepsilon$), que terminem com o evento a. Essa linguagem pode ser marcada por um autômato $G_2 = (X, E, f, \Gamma, x_0, X_m)$, cujo diagrama de transição de estados está representado na figura 3.3, em que $E = \{a, b\}$, $X = \{0, 1\}$, $x_0 = 0$, $X_m = \{1\}$ e f é definida da seguinte forma:

$$f(0, a) = 1 \qquad f(1, a) = 1 \tag{3.19}$$

$$f(0, b) = 0 \qquad f(1, b) = 0. \tag{3.20}$$

Note que, com a definição de f, a definição de Γ passa a ser redundante, uma vez que a partir de f é possível inferir o conjunto dos eventos ativos em cada estado do sistema. Nesse caso $\Gamma(x_i) = E$, $\forall x_i \in X$.

Quando todos os eventos são ativos em todos os estados do autômato, a função f do sistema é chamada de *função total*. Do contrário, a função é dita *parcial*.

É possível notar a partir da figura 3.3 que $\mathcal{L}_m(G_2) = L$. Além disso, o autômato

G_2, gera a seguinte linguagem:

$$\mathcal{L}(G_2) = \{\varepsilon, a, b, aa, ab, ba, bb, aaa, aab, aba, \ldots\}. \tag{3.21}$$

Note que $\mathcal{L}(G_2) = E^*$. Isso acontece sempre que a função f de um autômato é uma função total. Note também que ε aparece na linguagem gerada de G_2, apesar de não aparecer explicitamente no diagrama de transição de estados da figura 3.3.

3.3.2 Autômato com bloqueio

A partir da definição de linguagem gerada e linguagem marcada por um autômato G, é possível obter a seguinte relação de inclusão:

$$\mathcal{L}_m(G) \;\subseteq\; \overline{\mathcal{L}_m(G)} \;\subseteq\; \mathcal{L}(G). \tag{3.22}$$

A primeira relação se deve ao fato de X_m ser um subconjunto de X e a segunda inclusão é consequência da definição de $\mathcal{L}_m(G)$ e do fato de $\mathcal{L}(G)$ ser de prefixo fechado.

Um autômato G, no decorrer da sua evolução de estados, pode alcançar um estado $x_i \in X$, onde $\Gamma(x_i) = \emptyset$. Assim sendo, nenhum outro evento poderá ser executado e o sistema ficará estagnado. Se x_i pertencer ao conjunto dos estados marcados, X_m, isso pode indicar que o sistema completou seu objetivo ao atingir o estado x_i. No entanto, se $x_i \notin X_m$, essa situação indica que o sistema não pode mais alcançar qualquer estado pertencente ao conjunto X_m. Nesse caso, o estado x_i é considerado um *estado de bloqueio*.

Um autômato G possui um bloqueio quando em seu espaço de estados X existe um estado de bloqueio, ou seja, um estado a partir do qual não é possível alcançar um estado pertencente a X_m. Formalmente, um autômato G possui bloqueio se

$$\overline{\mathcal{L}_m(G)} \subset \mathcal{L}(G) \tag{3.23}$$

e G não possui bloqueio se:

$$\overline{\mathcal{L}_m(G)} = \mathcal{L}(G). \tag{3.24}$$

Existem dois tipo de bloqueio: (i) O bloqueio definitivo (*deadlock*), que se caracteriza quando um autômato alcança um determinado estado $x_i \notin X_m$, em que $\Gamma(x_i) = \emptyset$ e; (ii) O bloqueio ativo (*livelock*), que ocorre quando um autômato alcança um determinado estado $x_i \notin X_m$, em que $\Gamma(x_i) \neq \emptyset$ mas $f(x_i, s) \notin X_m$, para qualquer palavra $s \in E^*$.

Considere por exemplo o autômato G_3 ilustrado na figura 3.4, em que $E = \{a, b, c\}$, $X = \{0, 1, 2, 3, 4, 5\}$, $x_0 = 0$, $X_m = \{2\}$, f é definida como $f(0, a) = 1$,

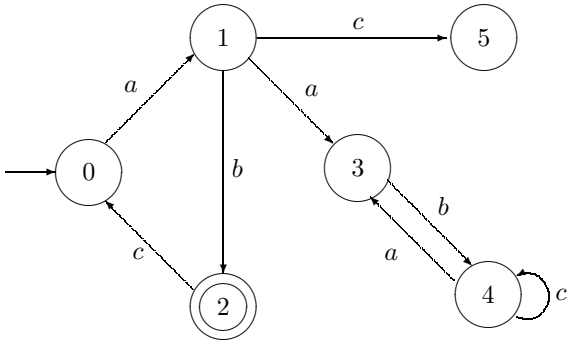


Figura 3.4: Autômato com bloqueio G_3.

$f(1,a) = 3,\ f(1,b) = 2,\ f(1,c) = 5,\ f(2,c) = 0,\ f(3,b) = 4,\ f(4,a) = 3,\ f(4,c) = 4$ e a linguagem gerada e marcada por G_3 é dada por:

$$\mathcal{L}(G_3) = \overline{\{abc\}^*} \cup \{a\}\{bca\}^*\{\{c\} \cup \{a\}\overline{\{\{b\}\{c\}^*\{a\}\}^*}\} \tag{3.25}$$

$$\mathcal{L}_m(G_3) = \{ab\}\{cab\}^*. \tag{3.26}$$

Note que, para melhor descrever a linguagem, foram usadas as notações de concatenação, fecho de prefixo e fecho de *Kleene* para linguagens. É fácil verificar que $\overline{\mathcal{L}_m(G_3)} \subset \mathcal{L}(G_3)$. Assim, o autômato G_3 possui bloqueios. Para este exemplo, o estado 5 é claramente um estado de bloqueio definitivo, pois $5 \notin X_m$ e $\Gamma(5) = \emptyset$. Além disso, os estados 3 e 4 são estados de bloqueio ativo, pois apesar de existirem transições partindo destes estados, é impossível alcançar um estado marcado a partir deles.

3.4 Operações com autômatos

Uma vez definida a forma de construção de um autômato e a sua relação com as linguagens que descrevem o comportamento de um SED, é preciso definir quais os tipos de operação admitidas por este modelo. Existem dois tipos básicos de operações com autômatos: as operações realizadas com um único autômato e as operações de composição, nas quais as manipulações ocorrem considerando as características de mais de um autômato. Essas operações são descritas a seguir.

3.4.1 Operações unárias

Nessas operações, o diagrama de transição de estados de um autômato é alterado considerando apenas as propriedades deste autômato. São elas: *acessibilidade*, *co-acessibililade* e *trim*. Estas operações sempre preservam o conjunto de eventos E do autômato.

Acessibilidade

Seja o autômato $G = (X, E, f, \Gamma, x_0, X_m)$, cuja linguagem marcada é $\mathcal{L}_m(G)$ e a linguagem gerada é $\mathcal{L}(G)$. Um estado $x_i \in X$ é acessível se existe uma palavra s pertencente a $\mathcal{L}(G)$ capaz de levar o sistema do estado x_0 até o estado x_i, ou seja $f(x_0, s) = x_i$. Do contrário, o estado é não acessível.

A parte acessível de um autômato G, pode ser obtida apagando do diagrama de transição os estados não-acessíveis de G e as transições que partem destes estados. Vale salientar que essa operação não altera nem a linguagem marcada, nem a linguagem gerada pelo autômato G, já que essas linguagens partem, por definição, do estado inicial x_0. Formalmente, a parte acessível de G é dada por:

$$Ac(G) := (X_{ac}, E, f_{ac}, x_0, X_{ac,m}), \tag{3.27}$$

em que

$$X_{ac} = \{x \in X : (\exists s \in E^*) \quad [f(x_0, s) = x]\}, \tag{3.28}$$

$$X_{ac,m} = X_m \cap X_{ac}, \tag{3.29}$$

$$f_{ac} = f|_{X_{ac} \times E \to X_{ac}}, \tag{3.30}$$

e a notação $f|_{X_{ac} \times E \to X_{ac}}$ indica que f_{ac} é igual à função f, só que restrita ao subconjunto de X formado apenas pelos estados acessíveis de G, X_{ac}. Quando todos os estados de um autômato forem acessíveis, isto é $G = Ac(G)$, este autômato será denominado *acessível*.

Co-acessibilidade

Seja o autômato $G = (X, E, f, \Gamma, x_0, X_m)$, cuja linguagem marcada é $\mathcal{L}_m(G)$ e a linguagem gerada é $\mathcal{L}(G)$. Um estado de G definido por $x_i \in X$ e $x_i \notin X_m$ é co-acessível, se existe uma palavra $s \in E^*$, capaz de levar o sistema do estado x_i até um estado $x_j \in X_m$, ou seja $f(x_i, s) = x_j \in X_m$. Do contrário o estado é não co-acessível.

A parte co-acessível de um autômato G, pode ser obtida apagando do diagrama de transição os estados não co-acessíveis de G e as transições que partem destes estados. Ao contrário da acessibilidade, a obtenção da parte co-acessível de G pode alterar a linguagem gerada pelo autômato, já a linguagem marcada por G permanece sem sofrer alterações. Formalmente, a parte co-acessível de G é dada por:

$$CoAc(G) := (X_{coac}, E, f_{coac}, x_{0,coac}, X_m), \tag{3.31}$$

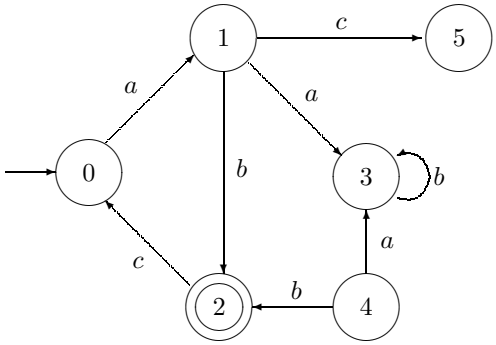


Figura 3.5: Autômato G_4.

em que

$$X_{coac} = \{x \in X : (\exists s \in E^*) \ \ [f(x,s) \in X_m]\} \tag{3.32}$$

$$x_{0,coac} = \begin{cases} x_0, & x_0 \in X_{coac} \\ \textit{não definido}, & \text{caso contrário} \end{cases} \tag{3.33}$$

$$f_{coac} = f|_{X_{coac} \times E \to X_{coac}}. \tag{3.34}$$

Quando todos os estados de um autômato forem co-acessíveis, isto é $G = CoAc(G)$, este autômato será denominado *co-acessível*.

Trim

A co-acessibilidade está estritamente relacionada com a idéia de bloqueio de um sistema. Dizer que um autômato G possui bloqueio significa dizer que $\mathcal{L}_m(G)$ é um subconjunto de $\mathcal{L}(G)$, onde $\mathcal{L}_m(G) \neq \mathcal{L}(G)$. Em outras palavras, um autômato possui bloqueio quando nele existem estados acessíveis, que não são co-acessíveis. Equivalentemente, um autômato não possui bloqueio quando todos os estados acessíveis são também co-acessíveis, e vice-versa. O autômato que é tanto acessível, quanto co-acessível é chamado de *trim*. Formalmente a operação *trim* é definida em G por:

$$Trim(G) := CoAc[Ac(G)] = Ac[CoAc(G)]. \tag{3.35}$$

Considere, por exemplo, o autômato G_4 ilustrado na figura 3.5, em que $E = \{a, b, c\}$, $X = \{0, 1, 2, 3, 4, 5\}$, $x_0 = 0$ e $X_m = \{2\}$. As linguagens gerada e marcada por G_4 são, respectivamente, dadas por:

$$\mathcal{L}(G_4) = \overline{\{abc\}^*} \cup \{a\}\{bca\}^*\{\{c\} \cup \{a\}\{b\}^*\} \tag{3.36}$$

$$\mathcal{L}_m(G_4) = \{ab\}\{cab\}^*. \tag{3.37}$$

O estado 4 é claramente o único estado não acessível de G_4. Ao aplicar a operação

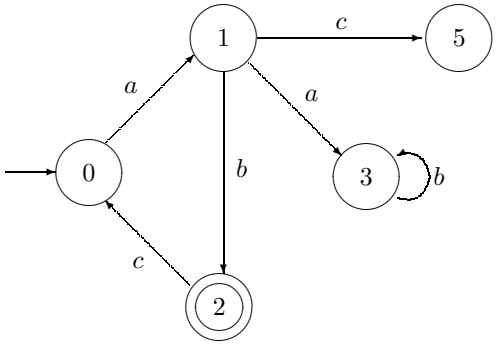


Figura 3.6: Acessibilidade de G_4, $Ac(G_4)$.

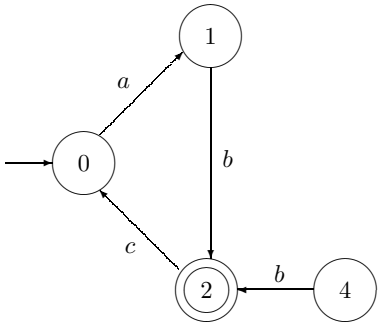


Figura 3.7: Co-acessibilidade de G_4, $CoAc(G_4)$.

de acessibilidade em G_4 o resultado é $Ac(G_4) = (X_{ac}, E, f_{ac}, x_0, X_{ac,m})$, em que $X_{ac} = \{0, 1, 2, 3, 5\}$, $X_{ac,m} = \{2\}$, cujo diagrama de transição de estados para $Ac(G_4)$ é apresentado na figura 3.6.

Em relação à co-acessibilidade de G_4, os estados 3 e 5 não são co-acessíveis. Aplicando em G_4 a operação de co-acessibilidade, obtém-se o seguinte resultado: $CoAc(G_4) = (X_{coac}, E, f_{coac}, x_{0,coac}, X_m)$, em que $X_{coac} = \{0, 1, 2, 4\}$, $x_{0,coac} = x_0 = 0$ e $X_{ac,m} = \{2\}$. A linguagem marcada permanece a mesma de G_4 e a linguagem gerada fica:

$$\mathcal{L}(CoAc[G_4]) = \overline{\{abc\}^*}. \tag{3.38}$$

O diagrama de transição de estados para $CoAc(G_4)$ é apresentado na figura 3.7.

Ainda em relação a G_4, é possível obter o autômato $trim(G_4)$ encontrando $Ac[CoAc(G_4)] = CoAc[Ac(G_4)]$. O resultado dessa operação é o autômato $trim(G_4) = (X_{trim}, E, f_{trim}, x_{0,trim}, X_{trim,m})$, em que $X_{trim} = \{0, 1, 2\}$, $x_{0,trim} = x_0 = 0$ e $X_{trim,m} = \{2\}$, cujo diagrama de transição de estados é apresentado na figura 3.8. Note que, conforme esperado,

$$\mathcal{L}[trim(G_4)] = \mathcal{L}[CoAc(G_4)] \tag{3.39}$$

$$\mathcal{L}_m[trim(G_4)] = \mathcal{L}_m[CoAc(G_4)]. \tag{3.40}$$

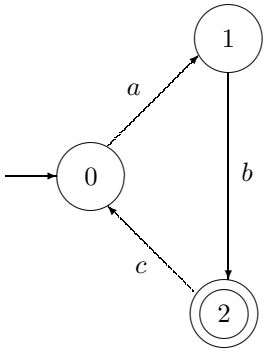


Figura 3.8: Trim de G_4, $trim(G_4)$.

3.4.2 Operações de composição

Nas operações de composição, o diagrama de transição de estados gerado é resultado da interação entre dois ou mais autômatos. As duas principais formas de composição de autômatos são a composição produto e a composição paralela. Essas operações são especialmente úteis quando se deseja descrever comportamentos complexos de um sistema a partir da composição de comportamentos mais simples.

A composição produto se caracteriza por permitir apenas transições associadas a *eventos comuns* (eventos que aparecem em todos os autômatos envolvidos na operação), o que restringe a sua utilização quando o interesse é modelar a interação entre comportamentos. Em geral, ao modelar sistemas que possuem componentes que interagem entre si, os eventos comuns capturam a ideia de acoplamento entre os modelos. Além dos eventos comuns, aparecem no modelo os *eventos privados*, que pertencem apenas a um dos modelos. A operação de composição padrão, capaz de criar o modelo de um sistema completo, através da interação entre os comportamentos individuais do sistema, preservando as transições privadas de cada comportamento é a chamada *composição paralela*.

Composição paralela

Considere dois autômatos:

$$G_1 = (X_1, E_1, f_1, \Gamma_1, x_{01}, X_{m1}) \quad e \quad G_2 = (X_2, E_2, f_2, \Gamma_2, x_{02}, X_{m2}). \tag{3.41}$$

Então, a composição paralela de G_1 e G_2 é definida como sendo:

$$G_1 \| G_2 := Ac(X_1 \times X_2, E_1 \cup E_2, f_{1\|2}, \Gamma_{1\|2}, X_{m1} \times X_{m2}), \tag{3.42}$$

em que $f_{1\|2}$ é definida por:

$$f_{1\|2}((x_1,x_2),e) := \begin{cases} (f_1(x_1,e),f_2(x_2,e)), & \text{se } e \in \Gamma_1(x_1) \cap \Gamma_2(x_2); \\ (f_1(x_1,e),x_2), & \text{se } e \in \Gamma_1(x_1) \setminus E_2; \\ (x_1,f_2(x_2,e)), & \text{se } e \in \Gamma_2(x_2) \setminus E_1; \\ n\tilde{a}o definida, & \text{caso contrário.} \end{cases} \tag{3.43}$$

$\Gamma_{1\|2}$ é dada por:

$$\Gamma_{1\|2}(x_1,x_2) = [\Gamma_1(x_1) \cap \Gamma_2(x_2)] \cup [\Gamma_1(x_1) \setminus E_2] \cup [\Gamma_2(x_2) \setminus E_1], \tag{3.44}$$

e as linguagens gerada e marcada por essa composição são, respectivamente:

$$\mathcal{L}(G_1\|G_2) = P_1^{-1}[\mathcal{L}(G_1)] \cap P_2^{-1}[\mathcal{L}(G_2)] \tag{3.45}$$

$$\mathcal{L}_m(G_1\|G_2) = P_1^{-1}[\mathcal{L}_m(G_1)] \cap P_2^{-1}[\mathcal{L}_m(G_2)], \tag{3.46}$$

em que P_i é a projeção de $E_1 \cup E_2$ em E_i, definida por [4] como sendo:

$$P_i : (E_1 \cup E_2)^* \longrightarrow E_i^*, \ parai = 1e2. \tag{3.47}$$

Os estados de $G_1\|G_2$ são denotados em pares, sendo a primeira componente o estado atual de G_1 e a segunda componente o estado atual de G_2. O estado (x_1,x_2) é marcado somente se $x_1 \in X_{m1}$ e $x_2 \in X_{m2}$.

No resultado da composição paralela, um evento $e \in E_1 \cap E_2$ só pode ser executado se ele ocorrer simultaneamente em todos os modelos envolvidos na composição. O que quer dizer que a composição é síncrona para *eventos comuns*. Dessa forma, os *eventos privados*, isto é, os eventos pertencentes ao conjunto $(E_1 \setminus E_2) \cup (E_2 \setminus E_1)$, não sofrem restrições e podem ser executados sempre que ocorrem em seus comportamentos de origem.

Considere como exemplo os autômatos G_1 (figura 3.2) e G_2 (figura 3.3). A composição paralela entre estes autômatos resulta em um novo autômato $G_1\|G_2$, em que:

$$X_{1\|2} = \{(x,0),(x,1),(y,0),(y,1),(z,0),(z,1)\} \tag{3.48}$$

$$E_{1\|2} = \{a,b,c\}, \tag{3.49}$$

sendo c o único evento privado, pertencente a G_1. O diagrama de transição de estado para $G_1\|G_2$ é aquele representado na figura 3.9.

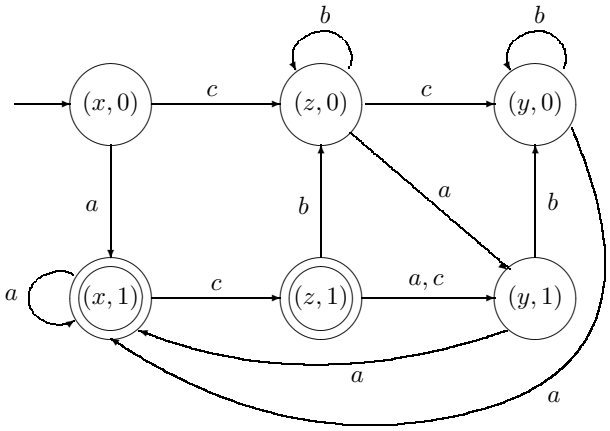


Figura 3.9: Composição paralela entre G_1 e G_2, $G_1 \| G_2$.

3.5 Modelo por eventos discretos de uma célula de manufatura

Para ilustrar a aplicação da teoria de SED, mais precisamente de autômatos, na modelagem de um sistema real, considere uma célula de manufatura formada por duas máquinas (M_1 e M_2) e um robô que transporta as peças de M_1 para M_2. A máquina M_1 recebe peças brutas e quando as peças estão prontas são recolhidas pelo robô. Caso o robô esteja ocupado, a máquina M_1 retém a peça até que o robô esteja completamente livre. Caso uma outra peça chegue enquanto a máquina M_1 estiver processando/retendo alguma peça, a máquina M_1 rejeita a peça recebida. Quando o robô recebe uma peça de M_1, inicia o transporte desta até a máquina M_2. No momento em que chegar a M_2, o robô somente entregará a peça à máquina M_2 se esta estiver livre; caso contrário reterá a peça até M_2 ficar disponível. Após entregar a peça a M_2, o robô retorna à máquina M_1. A máquina M_2 recebe a peça do robô e a processa.

A tabela 3.1 descreve os estados e os eventos das máquinas M_1 e M_2 e do robô. Note que os eventos e_1 (entrega de peça ao robô) e a_2 (entrega/chegada de peça em M_2) pertencem a dois subsistemas: máquina M_1 e robô, e robô e máquina M_2, respectivamente. É importante notar que, para que o evento e_1 ocorra, a máquina M_1 deverá estar no estado H_1 e o robô no estado I. Para que o evento a_2 ocorra, o robô deverá estar no estado H e a máquina M_2 deverá estar no estado I_2. Para os demais estados dos sistemas, isto é, aqueles que estão presentes em somente um dos subsistemas, a ocorrência não dependerá do estado em que os demais subsistemas estiverem, sendo determinada somente pelo estado atual do subsistema; por exemplo, a ocorrência do evento t_1 (fim de processamento da peça em M_1) dependerá apenas da máquina M_1 estar no estado P_1, independentemente de quais estados estiverem o robô e a máquina M_2.

Tabela 3.1: Os estados e os eventos das máquinas M_1, M_2 e do robô.

Elemento	Estados	Eventos
Máquina M_1	M_1 disponível:I_1 M_1 processando: P_1 M_1 retendo peça pronta: H_1 $X_1 = \{I_1, P_1, H_1\}$	Chegada de peça a M_1: a_1 Fim de processamento: t_1 Entrega de peça ao robô: e_1 $E_1 = \{a_1, t_1, e_1\}$
Robô	Robô disponível: I, Transportando $M_1 \rightarrow M_2$: T_{12} Esperando em M_2: H Retornando para M_1: R $X_r = \{I, T_{12}, H, R\}$	Entrega de peça ao robô: e_1 Chegada a M_2: c_2 Entrega/chegada de peça a M_2: a_2 Chegada a M_1: r_1 $E_r = \{e_1, c_2, a_2, r_1\}$
Máquina M_2	M_2 disponível: I_2 M_2 processando: P_2 $X_2 = \{I_2, P_2\}$	Entrega/chegada de peça em M_2: a_2 Fim de processamento: t_2 $E_2 = \{a_2, t_2\}$

Com o auxílio da tabela 3.1 é possível gerar os diagramas de transições dos autômatos das máquina M_1 e M_2, e do Robô, denominados por G_1 (figura 3.5.a), G_2 (figura 3.5.c) e G_r (figura 3.5.b), respectivamente.

O modelo do sistema pode ser obtido pela composição paralela de G_1, G_2 e G_r. Para fins didáticos, a análise da composição síncrona de somente dois subsistemas é mais interessante do que a da composição dos três subsistemas, visto que o autômato resultante possui muitos estados e transições. A figura 3.5 mostra a composição $G_r||G_2$. Note que o evento a_2 é um evento comum dos autômatos G_r e G_2 e esse evento somente poderá ocorrer caso G_r e G_2 estejam em estados cujos conjuntos dos eventos ativos tenham o evento a_2 como elemento. Note ainda que o evento a_2 não

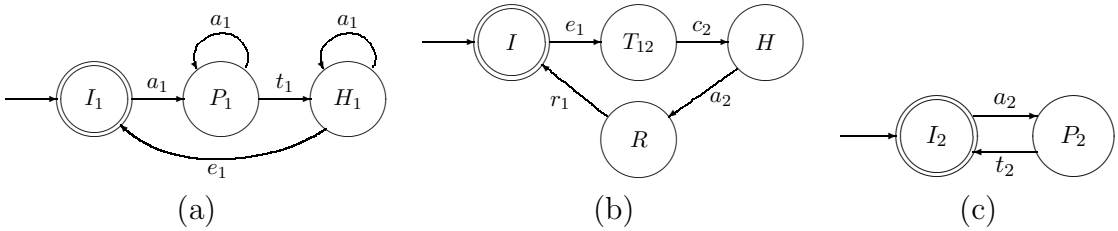


Figura 3.10: Diagrama de transição de estados: (a) G_1, da máquina M_1; (b) G_r, do robô; (c) G_2, da máquina M_2.

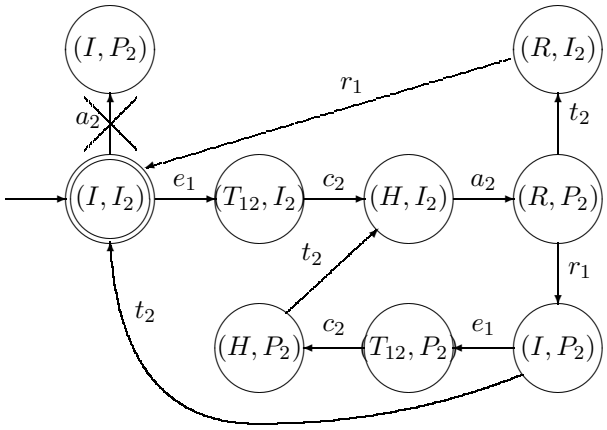


Figura 3.11: Diagrama de transição de estados da composição paralela entre G_r e G_2.

pertence ao conjunto dos eventos ativos do estado I de G_r e, portanto, não poderá ocorrer no estado (I, I_2) de $G_r||G_2$.

Capítulo 4

Controle Supervisório

Quando se realiza a modelagem de um sistema, na maioria da vezes, a intenção é entender melhor o seu funcionamento de modo a tornar possível o desenvolvimento de uma lei de controle capaz de modificar o comportamento desse sistema, para atingir uma certa especificação de desempenho.

Assim como nos sistemas modelados em espaço de estados contínuo, nos sistemas a eventos discretos a principal forma de se obter um comportamento específico desejado para o sistema é através de controle realimentado. Seja um SED modelado por um autômato G, sendo E é o conjunto de eventos de G. A premissa é que o comportamento de G descrito pela linguagem $\mathcal{L}(G)$, possui palavras, ou subpalavras, não desejáveis que violam certas condições que se deseja impor ao sistema. A essas condições dá-se o nome de *especificações*. Quando o funcionamento de um sistema não é satisfatório, ele deve ser modificado através de uma ação de controle, restringindo o comportamento desse sistema a um subconjunto de $\mathcal{L}(G)$. O sistema de controle que altera o comportamento de um SED com o intuito de atingir uma especificação desejada é chamado de *supervisor*, e é denotado pela letra S.

O fundamento da teoria de controle supervisório foi desenvolvido inicialmente por W. M. Wonham e P. J. Ramadge, e seus co-autores, nos anos 80 (veja [13] e suas referências). Desde então, diversos pesquisadores têm contribuído para o desenvolvimento e aplicação dessa vertente da teoria de controle em sistema a eventos discretos. Em um sistema de controle supervisório, S "observa" a ocorrência dos eventos em G e "diz" a G quais dos eventos ativos são permitidos. Mais precisamente, S possui a capacidade de desabilitar a ocorrência de certos eventos em G. A decisão sobre quais eventos desabilitar em G é atualizada sempre que S observa a ocorrência de um novo evento em G. Dessa forma, S exerce em G um *controle realimentado dinâmico*. No entanto, existem eventos em G cuja ocorrência não pode ser observada por S, como por exemplo a alteração na temperatura de uma planta onde não existe um sensor de temperatura. Esses eventos são chamados de *não-observáveis*. Neste trabalho todos os eventos considerados são observáveis. Existem também em G,

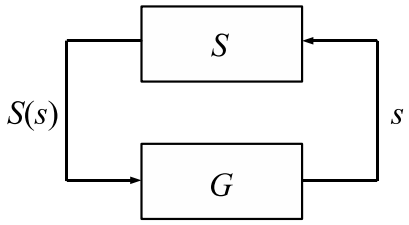


Figura 4.1: SED G e o supervisor realimentado S.

eventos cuja ocorrência não pode ser desabilitada devido à sua natureza, como, por exemplo, um evento que indica que um veículo autônomo colidiu. Esses eventos são chamados de *não-controláveis*.

Este capítulo está estruturado da seguinte forma. Na seção 4.1 são apresentadas os conceitos introdutórios sobre a teoria de controle supervisório. Na seção 4.2 é apresentado o conceito de controlabilidade de uma linguagem. Na seção 4.3 é apresentado o processo de modelagem de um especificação de controle através de um autômato. Na seção 4.4 é demonstrado o processo de realização de um sistema supervisor. Finalmente, na seção 4.5 é apresentada a formulação da medida de desempenho de uma linguagem μ.

4.1 Sistema supervisor S

Considere um SED, cujo comportamento é modelado pelo par de linguagens L e L_m, em que $L = \overline{L}$ é o conjunto de todas as sequências de eventos que o SED pode gerar e L_m é um subconjunto de L usado para representar o término de uma tarefa ou uma ação específica que se queira destacar, sendo definidas sobre um conjunto de eventos E. Sem perda de generalidade, pode-se dizer que L e L_m são as linguagens gerada e marcada pelo autômato

$$G = (X, E, f, \Gamma, x_0, X_m), \tag{4.1}$$

ou seja, $\mathcal{L}(G) = L$ e $\mathcal{L}_m(G) = L_m$. O problema do controle supervisório pode ser formulado da seguinte maneira: Dado um autômato G, obtenha um supervisor S para interagir com G de forma realimentada, como mostrado na figura 4.1, de tal forma que a linguagem gerada pelo sistema realimentado, denotada por $\mathcal{L}(S/G)$ (lê-se S controlando G), seja $\mathcal{L}(S/G) = L_a \subseteq L$ e $\mathcal{L}_m(S/G) = L_{m_a} = \mathcal{L}(S/G) \cap \mathcal{L}_m(G) = L_a \cap L_m$.

Considere, inicialmente, a seguinte partição do conjunto E:

$$E = E_c \dot{\cup} E_{uc}, \tag{4.2}$$

em que E_c é o conjunto dos eventos controláveis, E_{uc} é o conjunto dos eventos não-controláveis e $\dot{\cup}$ denota partição, isto é, $E_c \cap E_{uc} = \emptyset$ e $E_c \cup E_{uc} = E$. Supondo que todos os eventos em E são observáveis, então a função de transição de estados f, pode ser controlada por S dinamicamente, habilitando ou desabilitando em G eventos pertencentes a E_c.

Um supervisor S é uma *função* da linguagem gerada de G no conjunto potência de E, ou seja:

$$S : \mathcal{L}(G) \longrightarrow 2^E \tag{4.3}$$

$$s \in \mathcal{L}(G) \longmapsto \Gamma_N[f(x_0, s)] = S(s) \cap \Gamma[f(x_0, s)], \tag{4.4}$$

em que $\Gamma_N[f(x_0, s)]$ é o conjunto dos eventos que G pode executar no estado $f(x_0, s)$. Em outras palavras, G não pode executar um evento no estado $f(x_0, s)$ se este evento não pertencer a $S(s)$. Vale lembrar que, pela definição de eventos não-controláveis, os eventos em $\Gamma(f(x_0, s))$ pertencentes a E_{uc} não podem ser desabilitados, o que quer dizer que eles devem obrigatoriamente pertencer ao conjunto $S(s)$. Dessa forma, S é dito *admissível* se:

$$\forall s \in \mathcal{L}(G), \quad E_{uc} \cap \Gamma[f(x_0, s)] \subseteq S(s). \tag{4.5}$$

Um supervisor realizável deve ser admissível. Nessas condições, o conjunto $S(s)$ é a *ação de controle* do supervisor (ou controlador) S.

A linguagem gerada por S/G pode ser definida recursivamente da seguinte forma:

1. $\varepsilon \in \mathcal{L}(S/G)$;
2. $[(s \in \mathcal{L}(S/G))$ e $(s\sigma \in \mathcal{L}(G))$ e $(\sigma \in S(s))] \Leftrightarrow [s\sigma \in \mathcal{L}(S/G)]$,

em que

$$\mathcal{L}(S/G) \subseteq \mathcal{L}(G) \quad \text{e} \quad \mathcal{L}(S/G) = \overline{\mathcal{L}(S/G)}. \tag{4.6}$$

A linguagem marcada por S/G é dada por:

$$\mathcal{L}_m(S/G) := \mathcal{L}(S/G) \cap \mathcal{L}_m(G). \tag{4.7}$$

De uma forma geral, para um sistema controlado S/G, mantém-se a seguinte relação de inclusão:

$$\mathcal{L}_m(S/G) \subseteq \overline{\mathcal{L}_m(S/G)} \subseteq \mathcal{L}(S/G) \subseteq \mathcal{L}(G), \tag{4.8}$$

para quaisquer G e S.

4.2 Controlabilidade de uma linguagem

Considere o autômato $G = (X, E, f, \Gamma, x_0, X_m)$, para o qual $E_{uc} \subseteq E$ representa o conjunto dos eventos não-controláveis. Sejam K e $M = \overline{M}$ linguagens definidas sobre o conjunto de eventos E. Então, K é *controlável* com respeito a M e E_{uc} se:

$$\overline{K}E_{uc} \cap M \subseteq \overline{K}. \tag{4.9}$$

Por definição, a controlabilidade é uma propriedade do fecho do prefixo de uma linguagem, ou seja, K é controlável se, e somente se, $\overline{K}$ for controlável. A relação de controlabilidade pode ser interpretada da seguinte forma: para todo s pertencente a $\overline{K}$ e para todo e pertencente a E_{uc}, se K for controlável, então quando uma palavra *se* pertencer a M ela também pertencerá a $\overline{K}$.

Quando a linguagem K é controlável em relação à linguagem M e ao conjunto E_{uc}, então existe o supervisor S, cuja ação de controle $S(s)$ é dada por:

$$S(s) = [E_{uc} \cap \Gamma(f(x_0, s))] \cup \{\sigma \in E_c \cap \Gamma(f(x_0, s)) : s\sigma \in \overline{K}\}, \tag{4.10}$$

tal que $\mathcal{L}(S/G) = K$.

Uma maneira prática de se verificar a controlabilidade de K em relação a M e E_{uc}, quando $\mathcal{L}(K)$ e $\mathcal{L}(M)$ forem regulares, é através de uma simples comparação entre o autômato H (que gera a linguagem $\overline{K}$) e o autômato $H \times G$. A controlabilidade de K pode ser verificada comparando-se o conjunto de eventos ativos de cada estado de $H \times G$, com o conjunto de eventos ativos do estado correspondente em G (segunda componente do estado de $H \times G$). Se existir um evento ativo não-controlável em G, e este evento não estiver ativo no estado equivalente de $H \times G$, então a linguagem K será não controlável.

4.3 Modelagem de uma especificação

Para sintetizar este supervisor corretamente, no sentido de atender os requisitos especificados pela linguagem K, é preciso primeiramente representar essa especificação através do mesmo formalismo utilizado para representar o modelo do sistema a ser controlado, ou seja, através de um autômato. Dessa forma, é possível utilizar as operações entre autômatos para reproduzir a interação entre o supervisor S e o autômato G. Na prática essa operação é necessária, pois a maioria das especificações é dada na forma de instruções descritivas a respeito do comportamento de uma linguagem, como por exemplo só permitir a ocorrência do evento b depois que o evento a ocorrer, ou priorizar a ocorrência de uma dada sequência de eventos.

O processo de modelar uma especificação é o mesmo utilizado para modelar um

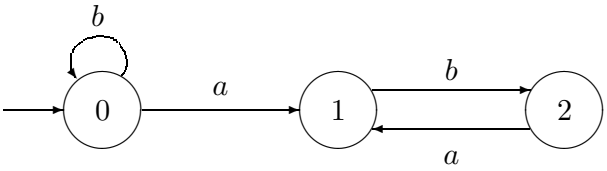


Figura 4.2: Autômato que modela a especificação 1.

sistema. Deve-se analisar o comportamento do sistema e representar da melhor forma possível esse comportamento dentro das limitações do formalismo utilizado na modelagem. Considere por exemplo as especificações 1 e 2, definidas abaixo:

- **Especificação 1**: Os evento a e b devem ocorrer de forma alternada, a partir da primeira ocorrência de a;
- **Especificação 2**: Os eventos a, b e c podem ocorrer livremente até que ocorra um evento d. Após a ocorrência de d, é obrigatória a execução da sequência $abca$ pelo sistema, antes que este volte a operar normalmente.

O diagrama de transição de estados do autômato que representa a linguagem da especificação 1 está ilustrado na figura 4.2. Analisando esse diagrama de transição de estados é possível notar que, antes da primeira ocorrência de a, o evento b pode ocorrer livremente, o que é caracterizado pelo *self-loop* no estado 0. Depois que a transição a é executada pela primeira vez, o sistema passa para o estado 1, onde a única transição possível é aquela associada ao evento b, que leva o sistema ao estado 2. No estado 2, o único evento que pode ocorrer é o evento a, que leva o sistema novamente para o estado 1, caracterizando a idéia de eventos alternados.

O diagrama de transição de estados do autômato que representa a linguagem da especificação 2 está ilustrado na figura 4.3. Analisando o diagrama de transição de estados é possível notar que os eventos a, b e c podem ocorrer livremente, até que ocorra um evento d, que leva o sistema para o estado 1. Uma vez no estado 1, a única sequência de eventos possível é $abca$, que faz o sistema passar pelos estados 2, 3 e 4 em sequência, para depois retornar ao estado 0, onde a execução dos eventos a, b e c é livre, até que ocorra novamente o evento d, cumprindo as exigências da especificação.

4.4 Realização de um supervisor

Uma vez construído o autômato que modela a especificação $K \subset \mathcal{L}(G)$, é preciso construir uma representação conveniente para a função S, cuja ação de controle $S(s)$ é definida na equação (4.10). Isso se deve ao fato de que não seria prático listar $S(s)$ para toda palavra $s \in \mathcal{L}(G)$.

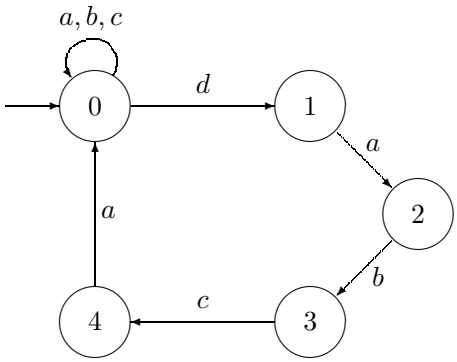


Figura 4.3: Autômato que modela a especificação 2.

Quando a linguagem gerada pela especificação K, assim como a linguagem do sistema G, for regular, então a linguagem de S/G será também regular, portanto realizável. Nesse caso, é possível utilizar a composição paralela entre os autômatos que modelam o sistema e a especificação, como uma forma algébrica de capturar o efeito de S controlando G. Quando K for uma linguagem controlável, essa realização do supervisor S é chamada de *realização padrão do supervisor*.

Seja G o autômato que modela o sistema e H o autômato que modela a especificação K, isto é,

$$\mathcal{L}(H) = \overline{K} = K. \tag{4.11}$$

Então, o comportamento desejado para o sistema controlado S/G será exatamente o comportamento capturado pela composição paralela $H\|G$, cuja linguagem gerada é

$$\begin{aligned} \mathcal{L}(H\|G) &= \mathcal{L}(H) \cap \mathcal{L}(G) && (4.12)\\ &= \overline{K} \cap \mathcal{L}(G) && (4.13)\\ &= \overline{K} = \mathcal{L}(S/G), && (4.14) \end{aligned}$$

cuja linguagem marcada é:

$$\begin{aligned} \mathcal{L}_m(H\|G) &= \mathcal{L}_m(H) \cap \mathcal{L}_m(G) && (4.15)\\ &= \overline{K} \cap \mathcal{L}_m(G) && (4.16)\\ &= \mathcal{L}(S/G) \cap \mathcal{L}_m(G) = \mathcal{L}_m(S/G). && (4.17) \end{aligned}$$

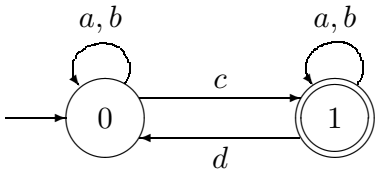


Figura 4.4: Modelo da planta G a ser controlada.

Utilizando a realização padrão para o supervisor S, a ação de controle é:

$$\begin{aligned} S(s) &= [E_{uc} \cap \Gamma(f(x_0, s))] \cup \{\sigma \in E_c \cap \Gamma(f(x_0, s)) : s\sigma \in \overline{K}\} \\ &= \Gamma_H[f_H(x_{H0}, s)] \\ &= \Gamma_{H\|G}[f_H \| f_G((x_{H0}, x_{G0}), s)], \end{aligned} \tag{4.18}$$

em que a primeira igualdade é a mesma da equação (4.10), a segunda é proveniente do fato de que $\overline{K} \subseteq \mathcal{L}(G)$ e a terceira igualdade representa o conjunto dos eventos ativos da composição paralela ($\Gamma_{H\|G}$), sendo $f_H \| f_G$ a notação que indica a função de transição da composição $H\|G$. Dessa forma é possível construir uma realização finita para o sistema controlado S/G.

Para ilustrar o processo de representação de um sistema controlado S/G, considere a planta G cujo diagrama de transição de estados é apresentado na figura 4.4, com $G = (X, E, f, \Gamma, x_0, X_m)$ e $E = E_{uc} \cup E_c$, sendo $X = \{0, 1\}$, $E = \{a, b, c, d\}$, $x_0 = 0$, $E_{uc} = \{c, d\}$, $E_c = \{a, b\}$ e $X_m = 1$. As linguagens gerada e marcada pela planta são, respectivamente:

$$\mathcal{L}(G) = \overline{\{\{a, b\}^* c \{a, b\}^* d\}^*} \tag{4.19}$$

$$\mathcal{L}_m(G) = \{a, b\}^* c \{a, b\}^* \{d \{a, b\}^* c \{a, b\}^*\}^*. \tag{4.20}$$

A especificação que se deseja atender é que os evento a e b ocorram de forma alternada, sendo que o evento a deve ser o primeiro a ocorrer. Essa especificação pode ser modelada pelo autômato H, cujo diagrama de transição de estados está ilustrado na figura 4.5, onde $H = (X_H, E_H, f_H, \Gamma_H, x_H 0, X_H m)$ e $X_{Hm} = X_H$. A linguagem gerada pelo autômato H é:

$$\mathcal{L}(H) = \mathcal{L}_m(H) = \overline{\{ab\}^*}. \tag{4.21}$$

Normalmente os estados de uma especificação são todos marcados, com o intuito de manter como estados marcados do sistema controlado aqueles marcados originalmente no modelo da planta. No entanto, não existe qualquer exigência a respeito dos estados marcados de uma especificação. Deve-se somente lembrar que, o fato de haver estados não marcados na especificação pode alterar a marcação original da

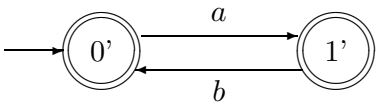


Figura 4.5: Modelo da especificação H.

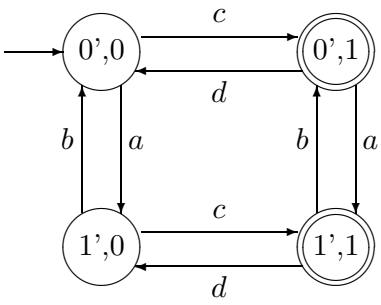


Figura 4.6: Modelo do sistema controlado S/G ($H \| G$).

planta no sistema controlado, o que pode vir a ser útil a depender do objetivo da especificação e do significado atribuído pelo projetista à marcação.

Uma vez que E_H possui apenas eventos controláveis, $\mathcal{L}(H)$ é obrigatoriamente controlável, o que possibilita a aplicação da realização padrão de um supervisor através da composição paralela entre H e G, sendo $\mathcal{L}(S/G) = \mathcal{L}(H \| G)$. O diagrama de transição de estados do sistema controlado é apresentado na figura 4.6.

Comparando os diagramas de transição de estados da planta (figura 4.4) e do sistema controlado (figura 4.6, é possível notar que a diferença entre o sistema controlado e a planta original é simplesmente a ocorrência alternada de a e b no sistema controlado, sendo a ocorrência de b possível somente após a primeira ocorrência de a. Pode-se concluir assim que o sistema S/G modela exatamente o comportamento desejado.

A controlabilidade da linguagem $\mathcal{L}(S/G)$ em relação ao conjunto dos eventos não controláveis, $E_{uc} = \{c, d\}$, é facilmente verificada através do teste apresentado na seção 4.2, em que todo evento não controlável ativo em G também está ativo em $H \| G$, garantindo, assim, a controlabilidade da linguagem $\mathcal{L}(S/G)$, o que faz do sistema controlado um sistema admissível.

É importante ressaltar que, apesar dessa seção apresentar a realização de um supervisor, não foi abordado o tema da máxima linguagem controlável de uma especificação K [4]. Isso se deve ao fato de que optou-se por apresentar nesse capítulo, somente a fundamentação teórica que será utilizada no desenvolvimento desse trabalho. Uma vez que a linguagem da especificação desenvolvida no capítulo 5 é controlável, não foi necessária a apresentação de técnicas de obtenção de supervisores para o caso da linguagem da especificação ser parcialmente controlável.

4.5 Índice de desempenho μ

Uma especificação K é uma informação qualitativa a respeito de um comportamento a partir da qual se elabora um modelo que a represente (ver seção 4.3). No entanto, o processo de construção de um autômato para representar uma dada especificação não é único, ou seja, uma mesma especificação pode ser representada por diferentes autômatos que geram a mesma linguagem, ou mesmo linguagens distintas, que atendam à mesma especificação. Dessa forma, a realização padrão de um controle supervisório S/G, obtida a partir da composição paralela $H\|G$, em que $\mathcal{L}(H) = K = \overline{K}$ e G é o modelo da planta, também não é única. Nesse contexto, é de fundamental importância a utilização de uma métrica formal capaz de avaliar o desempenho de diferentes controladores, possibilitando, assim, a escolha do supervisor que melhor realize uma dada especificação.

Tradicionalmente, o conceito de permissividade tem sido utilizado em controle supervisório (ver [4], [46] e suas referências) para facilitar uma comparação qualitativa entre controladores desenvolvidos sob uma mesma condição de controlabilidade de linguagem. Supervisores de máxima permissividade têm sido propostos por diversos pesquisadores baseados em diferentes condições de operação [14]. No entanto, máxima permissividade não implica, necessariamente, um melhor desempenho do sistema supervisionado [47]. Por exemplo, no problema do caixeiro viajante, um supervisor de máxima permissividade pode não encontrar o caminho mais curto para visitar as cidades e retornar ao ponto de partida, uma vez que nenhuma medida quantitativa de desempenho é utilizada no desenvolvimento desse tipo de supervisor [47].

O problema apresentado serviu de motivação para se propor uma *medida real para linguagens regulares* (μ), ou simplesmente *medida de linguagem* (ver [16], [48] e [15]), que seja capaz de avaliar e comparar quantitativamente o desempenho de diferentes controladores sem a necessidade de utilizar o conceito de permissividade. Na construção da medida de linguagem μ, cada estado marcado é caracterizado por um valor real, positivo ou negativo, que é escolhido com base na percepção do projetista a respeito do impacto do estado no desempenho do sistema. Conceitualmente similar à probabilidade condicional, a cada evento é associado um custo baseado no estado em que este evento é gerado. Esse procedimento permite associar a uma sequência de eventos que termine em um estado marcado bom (ou ruim), um valor real positivo (ou negativo). Dessa forma, um supervisor pode ser desenvolvido no sentido de eliminar da linguagem controlada o maior número possível de palavras indesejadas e manter o menor número possível de palavras desejadas.

Diferentes supervisores podem atingir uma especificação de diferentes formas, gerando assim um conjunto não ordenado (ou possivelmente parcialmente ordenado)

de linguagens controladas. Dessa forma, utilizando um critério de desempenho, a medida de linguagem μ possibilita uma ordenação total dessas linguagens, permitindo uma comparação quantitativa do comportamento da planta quando controlada por diferentes supervisores. Uma grande vantagem da utilização da medida de linguagem μ é que esse índice pode ser definido independentemente de como o supervisor foi concebido, seja por máxima permissividade, supervisor com bloqueio, sem bloqueio ou qualquer outra forma de projeto utilizada.

4.5.1 Formulação da medida de linguagem μ

Seja $G = (X, E, f, \Gamma, x_0, X_m)$ o autômato que modela o comportamento de uma planta e seja n a cardinalidade de X, isto é, $|X| = n$. Suponha que $\mathcal{L}(G)$ e $\mathcal{L}_m(G)$ denotem, respectivamente, as linguagens gerada e marcada pelo autômato G. Seja $\mathcal{L}(G_i)$ a linguagem gerada pelo autômato G, a partir do estado x_i, ou seja:

$$\mathcal{L}(G_i) = \{s \in E^* : f(x_i, s) \in X\}, \tag{4.22}$$

e $\mathcal{L}_m(G_i)$ a linguagem marcada pelo autômato G a partir do estado x_i, ou seja:

$$\mathcal{L}_m(G_i) = \{s \in E^* : f(x_i, s) \in X_m\}. \tag{4.23}$$

Particione o conjunto de estados marcados de G, X_m, em dois subconjuntos X_m^+ e X_m^-, isto é:

$$X_m = X_m^+ \dot{\cup} X_m^-, \tag{4.24}$$

em que X_m^+ contém todos os estados marcados bons (que se deseja alcançar) e X_m^- contém todos os ruins (que se deseja evitar). Em geral, a linguagem $\mathcal{L}_m(G_i)$ para $i = 0$, possui sequências de eventos que levam tanto a estados em X_m^+ quanto a estados em X_m^-. A sublinguagem de $\mathcal{L}(G_i)$ composta pelas sequências que não pertencem à linguagem marcada $\mathcal{L}_m(G_i)$ é denotada por $\mathcal{L}_i$, isto é:

$$\mathcal{L}_i = \{s \in \mathcal{L}(G_i) : f(x_i, s) \notin X_m\}. \tag{4.25}$$

Uma explicação mais detalhada a respeito da partição dos estados marcados em X_m^+ e X_m^- pode ser encontrada em [15] e [49]. Suponha ainda que $\mathcal{L}(x_i, x_k)$ denote o conjunto de todas as palavras que começam no estado x_i e terminam no estado $x_k \in X$, ou seja:

$$\mathcal{L}(x_i, x_k) = \{s \in \mathcal{L}(G_i) : (\exists x_k \in X)[f(x_i, s) = x_k]\}. \tag{4.26}$$

Dessa forma, as linguagens regulares $\mathcal{L}(G_i)$ e $\mathcal{L}_m(G_i)$ podem ser expressas da seguinte forma:

$$\mathcal{L}(G_i) = \bigcup_{x_k \in X} \mathcal{L}(x_i, x_k) = \bigcup_{k=0}^{n-1} \mathcal{L}(x_i, x_k) \tag{4.27}$$

$$\mathcal{L}_m(G_i) = \bigcup_{x_k \in X_m} \mathcal{L}(x_i, x_k) = \mathcal{L}_m^+ \cup \mathcal{L}_m^-, \tag{4.28}$$

sendo $\mathcal{L}_m^+$ e $\mathcal{L}_m^-$ as sublinguagens boas e ruins de $\mathcal{L}_m(G_i)$, respectivamente. Note que, para um dado x_i, a linguagem $\mathcal{L}(x_i, x_k) \in \mathcal{L}(G_i)$ é unicamente definida para um estado terminal x_k, com $k \in \{0, 1, \ldots, n-1\}$, ou seja:

$$\mathcal{L}(x_i, x_k) \cap \mathcal{L}(x_i, x_j) = \emptyset, \forall j \neq k. \tag{4.29}$$

Consequentemente:

$$\mathcal{L}(G_i) = \mathcal{L}_i \cup \mathcal{L}_m^+ \cup \mathcal{L}_m^-, \tag{4.30}$$

em que:

$$\mathcal{L}_i = \bigcup_{x \notin X_m} \mathcal{L}(x_i, x). \tag{4.31}$$

Suponha, agora, que aos estados em X_m^+ sejam associados pesos positivos, aos estados em X_m^- sejam associados pesos negativos e aos demais estados peso nulo. Dessa forma, define-se a *função característica* $\mathcal{X}$ como:

$$\mathcal{X} : X \longrightarrow [-1, 1] \tag{4.32}$$

$$x \longmapsto \mathcal{X}(x) \begin{cases} \in [-1, 0), & x \in X_m^-; \\ = 0, & x \notin X_m; \\ \in (0, 1], & x \in X_m^+. \end{cases} \tag{4.33}$$

O vetor cujos elementos são as funções características $\mathcal{X}(x_i) = \mathcal{X}_i$, é chamado de *vetor característico* e é denotado por $\underline{\mathcal{X}} = [\mathcal{X}_0 \mathcal{X}_1 \cdots \mathcal{X}_{n-1}]^T$. Dessa forma, o *k-ésimo* elemento do vetor $\underline{\mathcal{X}}$ é o peso associado ao estado correspondente x_k.

Considere, agora, o problema de se determinar a probabilidade de ocorrência de cada sequência $s \in \mathcal{L}(x_i, x_k)$. Para determinar a probabilidade de ocorrência de s, é preciso conhecer antes a probabilidade de ocorrência (custo) de cada evento e pertencente s. Para tanto, será suposto que o modelo é Markov, isto é, a probabilidade condicional presente, passada e futura são independentes [16]. Dessa forma, a probabilidade de ocorrência de um evento é conceitualmente similar à probabilidade condicional de transição do modelo, onde se considera o estado atual e as transições ativas neste estado para determinar o custo do evento. Isto leva à seguinte definição.

Definição 4.1 *(Probabilidade de ocorrência de um evento em G) A probabilidade*

de ocorrência de um evento em G, é definida pela função:

$$\begin{aligned} \tilde{\pi} : E^* \times X &\longrightarrow [0,1] \\ (s,x) &\longmapsto \tilde{\pi}(s,x), \end{aligned} \tag{4.34}$$

satisfazendo às seguintes condições:

1. $\tilde{\pi}(e_j, x_i) \begin{cases} \tilde{\pi}_{ij} \in (0,1), & \forall e_j \in \Gamma(x_i) \\ 0, & \forall e_j \notin \Gamma(x_i); \end{cases}$
2. $\sum_j \tilde{\pi}_{ij} < 1, \quad \forall x_i \in X;$
3. $\tilde{\pi}(e_j s, x_i) = \tilde{\pi}(e_j, x_i)\tilde{\pi}(s, f(x_i, e_j))$.

Para a introdução de uma medida de linguagem as seguintes definições são necessárias:

Definição 4.2 *A medida $\mu_i(s)$ de uma dada sequência de eventos s, em que $f(x_i, s) = x_k$, iniciada no estado x_i, é definida em função de $\tilde{\pi}$ e da função característica $\mathcal{X}_k$ da seguinte forma:*

$$\mu_i(s) := \tilde{\pi}[s, x_i]\mathcal{X}_k. \tag{4.35}$$

De acordo com a definição acima, tem-se que:

$$\forall s \in \mathcal{L}(x_i, x_k), \qquad \mu_i(s) \begin{cases} = 0, & x_k \notin X_m; \\ > 0, & x_k \in X_m^+; \\ < 0, & x_k \in X_m^-. \end{cases} \tag{4.36}$$

Definição 4.3 *A medida μ_i de uma dada linguagem $\mathcal{L}(x_i, x_k)$, para um dado estado $x_k \in X$, é definida por:*

$$\mu_i(\mathcal{L}(x_i, x_k)) = \left(\sum_{s \in \mathcal{L}(x_i, x_k)} \tilde{\pi}[s, x_i] \right) \mathcal{X}_k. \tag{4.37}$$

Com base nas definições 4.2 e 4.3, é possível introduzir a seguinte medida de linguagem:

Definição 4.4 *(Medida de linguagem μ_i) A medida de linguagem μ_i da linguagem $\mathcal{L}(G_i)$, é definida da seguinte forma [16, 47]:*

$$\mu_i := \sum_{x \in X} \mu_i(\mathcal{L}(x_i, x)) + \mathcal{X}_i. \tag{4.38}$$

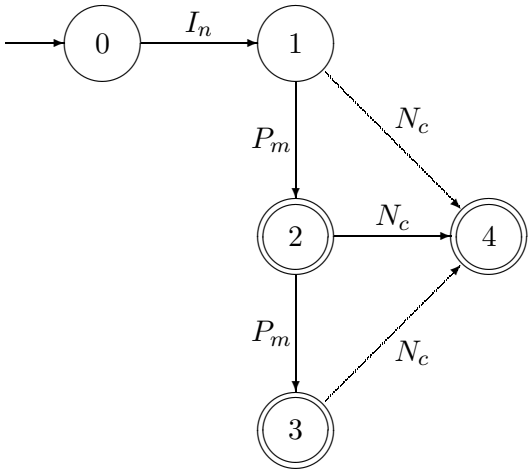


Figura 4.7: Autômato de navegação G_n.

Tabela 4.1: Descrição dos eventos e estados do autômato G_n, ilustrado na figura 4.7.

Evento/Estado	Descrição
Evento I_n	*Iniciar a navegação*
Evento P_m	*Problema detectado em um motor*
Evento N_c	*Navegação concluída*
Estado 0	*Disponível*
Estado 1	*Navegando*
Estado 2	*Navegando com problemas em um motor*
Estado 3	*Navegando com problemas nos dois motores*
Estado 4	*Concluiu a tarefa*

Observação 4.1 *O vetor $\boldsymbol{\mu}$, cujos elementos são os valores de μ_i, $i = 0, 1, \ldots, n-1$, é chamado de vetor de medida de linguagem ou vetor-$\boldsymbol{\mu}$, cujo i-ésimo elemento desse vetor representa a medida de linguagem de todas as sequências que partem do estado x_i para qualquer outro estado do autômato analisado.*

Considere como exemplo o autômato G_n, ilustrado na figura 4.7, que representa o modelo em SED de um sistema hipotético de navegação de um robô móvel, no qual existem dois atuadores responsáveis pela movimentação deste robô. O conjunto dos eventos e o espaço de estados desse autômato são $E = \{I_n, P_m, N_c\}$ e $X = \{0, 1, 2, 3, 4\}$, respectivamente, cujos significados são apresentados na tabela 4.1. O autômato G_n modela apenas o comando de iniciar a navegação (evento I_n) e indica quando um dos motores apresentou algum problema (evento P_m) ou quando a navegação foi concluída (evento N_c).

Inicialmente é preciso definir o vetor característico $\underline{\mathcal{X}}$ desse sistema. Considerando os estados 2 e 3 como estados marcados ruins (pois eles indicam que um motor

apresentou algum problema) e o estado 4 como estado marcado bom (pois indica que o robô concluiu a tarefa), o vetor característico deve ser da seguinte forma:

$$\mathcal{X}_i \begin{cases} = 0, & \text{se } i = \{0, 1\}; \\ < 0, & \text{se } i = \{2, 3\}; \\ > 0, & \text{se } i = 4. \end{cases} \tag{4.39}$$

Para esse exemplo, os valores escolhidos foram:

$$\underline{\mathcal{X}}^T = \begin{bmatrix} 0 & 0 & -0,1 & -0,2 & 1 \end{bmatrix}. \tag{4.40}$$

Definido o vetor característico, é preciso definir a probabilidade de ocorrência de um evento $\tilde{\pi}_{ij}$, para todo $e_j \in E$ e todo $x_i \in X$. Como as probabilidades $\tilde{\pi}_{ij}$ são desconhecidas, será utilizado, para fins de simplificação, $\tilde{\pi}_{ij} = 0,3$ para todos os valores admissíveis de i e j. Com isso, é possível calcular o valor de todos os elementos do vetor $\boldsymbol{\mu}$, utilizando a equação (4.38). Assim, o elemento μ_0, por exemplo, é dado por:

$$\mu_0 = \sum_{x \in X} \mu_0(\mathcal{L}(0, x)) + \mathcal{X}_0 \tag{4.41}$$

$$\mu_0 = \mu_0(\mathcal{L}(0,1)) + \mu_0(\mathcal{L}(0,2)) + \mu_0(\mathcal{L}(0,3)) + \mu_0(\mathcal{L}(0,4)) + \mathcal{X}_0. \tag{4.42}$$

Calculando as parcelas do lado direito da equação (4.42), de acordo com a equação (4.37), obtém-se:

$$\mu_0(\mathcal{L}(0,1)) = (\tilde{\pi}(I_n, 0))\mathcal{X}_1 = (0,3)(0) = 0 \tag{4.43}$$

$$\mu_0(\mathcal{L}(0,2)) = (\tilde{\pi}(\{I_n P_m\}, 0))\mathcal{X}_2 = (0,3^2)(-0,1) = -0,009 \tag{4.44}$$

$$\mu_0(\mathcal{L}(0,3)) = (\tilde{\pi}(\{I_n P_m P_m\}, 0))\mathcal{X}_3 = (0,3^3)(-0,2) = -0,0054 \tag{4.45}$$

$$\mu_0(\mathcal{L}(0,4)) = (\tilde{\pi}(\{I_n N_c\}, 0) + \tilde{\pi}(\{I_n P_m N_c\}, 0) + \tilde{\pi}(\{I_n P_m P_m N_c\}, 0))\mathcal{X}_4 \tag{4.46}$$

$$= (0,3^2 + 0,3^3 + 0,3^4)(1) = 0,1251 \tag{4.47}$$

$$\mathcal{X}_0 = 0. \tag{4.48}$$

Reescrevendo a equação (4.42) com os valores calculados acima, tem-se:

$$\mu_0 = 0 - 0,009 - 0,0054 + 0,1251 + 0 = 0,1107. \tag{4.49}$$

Repetindo o cálculo apresentado acima para os demais estados do autômato G_n,

obtém-se os seguintes valores:

$$\mu_1 = 0,369 \tag{4.50}$$
$$\mu_2 = 0,230 \tag{4.51}$$
$$\mu_3 = 0,100 \tag{4.52}$$
$$\mu_4 = 1,000. \tag{4.53}$$

Assim, o vetor de medida de linguagem, para o autômato G_n, é $\boldsymbol{\mu}^T = \begin{bmatrix} 0,1107 & 0,3690 & 0,2300 & 0,1000 & 1,000 \end{bmatrix}$.

4.5.2 Cálculo matricial de μ para S_t/G

Um supervisor S_t qualquer define uma linguagem admissível $\mathcal{L}(S_t/G)$ para o sistema, sendo esta linguagem um subconjunto de $\mathcal{L}(G)$. Diferentes supervisores, $S_t : t = 1, 2, \ldots, m$, podem ser projetados para se atingir um determinado objetivo de diferentes formas, gerando assim um conjunto parcialmente ordenado de sublinguagens de $\mathcal{L}(G)$, $L_t = \{\mathcal{L}(S_t/G) : t = 1, 2, \ldots, m\}$. Utilizando o procedimento apresentado na seção 4.5.1, é possível obter o valor de μ_0 para cada linguagem controlada $\mathcal{L}(S_t/G)$, dado que $\mathcal{L}(S_t/G) \subseteq \mathcal{L}(G)$, possibilitando assim a escolha do supervisor que apresenta o melhor desempenho, ou seja, o maior valor de μ_0.

Seja $\widetilde{\mathbf{\Pi}}$ a representação matricial de todas as probabilidades de ocorrência de eventos existentes em G, $\tilde{\pi}_{ij}$, para qualquer $e_j \in E$, com $j = 1, 2, \ldots, l$, e qualquer $x_i \in X$, com $i = 0, 1, \ldots, n-1$. Dessa forma a matriz $\widetilde{\mathbf{\Pi}}$ pode ser definida como:

$$\widetilde{\mathbf{\Pi}} = \begin{bmatrix} \tilde{\pi}_{11} & \tilde{\pi}_{12} & \cdots & \tilde{\pi}_{1l} \\ \tilde{\pi}_{21} & \tilde{\pi}_{21} & \cdots & \tilde{\pi}_{2l} \\ \vdots & \vdots & \ddots & \vdots \\ \tilde{\pi}_{n1} & \tilde{\pi}_{n2} & \cdots & \tilde{\pi}_{nl} \end{bmatrix}. \tag{4.54}$$

A matriz $\widetilde{\mathbf{\Pi}}$ define as probabilidades de ocorrência de cada evento em um dado estado de G, independentemente da linguagem a ser executada pelo sistema controlado. Dado um autômato G_t, cuja linguagem seja definida como:

$$\mathcal{L}(G_t) = \mathcal{L}(S_t/G) \subseteq \mathcal{L}(G), \tag{4.55}$$

é possível construir, a partir de $\widetilde{\mathbf{\Pi}}$, uma matriz cujos elementos representam somente as transições de estado permitidas em $\mathcal{L}(S_t/G)$. Essa matriz recebe o nome de *matriz*

de transição de estados, sendo definida da seguinte forma:

$$\mathbf{\Pi} = \begin{bmatrix} \pi_{11} & \pi_{12} & \cdots & \pi_{1n} \\ \pi_{21} & \pi_{21} & \cdots & \pi_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ \pi_{n1} & \pi_{n2} & \cdots & \pi_{nn} \end{bmatrix}, \tag{4.56}$$

em que o elemento π_{ij} da matriz $\mathbf{\Pi}$ representa a probabilidade de ocorrência de uma transição do estado x_i para o estado x_j. O valor de cada elemento π_{ij} pode ser obtido, para quaisquer $\{x_i, x_j\} \in X$ e qualquer $e \in E$, a partir dos valores de probabilidade de ocorrência de um evento $\tilde{\pi}$, sendo $\pi : X \times X \rightarrow [0, 1)$. Assim, o elemento π_{ij} é definido da seguinte forma:

$$\pi_{ij} = \begin{cases} \sum_{e \in E} \tilde{\pi}(e, x_i), & \text{se } f(x_i, e) = x_j; \\ 0, & \text{se } \{f(x_i, e) = x_j\} = \emptyset. \end{cases} \tag{4.57}$$

Reescrevendo a equação (4.38), em função de π e $\mathcal{X}$ (ver [16] e [47]), obtém-se:

$$\mu_i = \sum_j \pi_{ij} \mu_j + \mathcal{X}_i, \tag{4.58}$$

que em notação vetorial pode ser escrita como:

$$\boldsymbol{\mu} = \mathbf{\Pi}\boldsymbol{\mu} + \underline{\mathcal{X}}, \tag{4.59}$$

em que cada elemento μ_i do vetor $\boldsymbol{\mu}$ é a medida de linguagem para o modelo do sistema quando este é iniciado a partir do estado x_i. A solução para a equação (4.59) é dada por:

$$\boldsymbol{\mu} = (\mathbf{I} - \mathbf{\Pi})^{-1} \underline{\mathcal{X}}, \tag{4.60}$$

uma vez que a matriz $\mathbf{I} - \mathbf{\Pi}$ é não singular [47]. Dessa forma, para calcular os elementos de $\boldsymbol{\mu}$ para qualquer subconjunto de $\mathcal{L}(G)$, basta determinar o vetor característico, $\underline{\mathcal{X}}$, e a matriz de transição de estados, $\mathbf{\Pi}$, que define as probabilidades de ocorrência de todas as possíveis transições.

Para ilustrar os conceitos revisados nessa seção, considere novamente o autômato G_n, ilustrado na figura 4.7. Utilizando a definição de matriz de transição de estados, $\mathbf{\Pi}$, e a equação (4.60), é possível obter o mesmo vetor $\boldsymbol{\mu}$ de forma mais simples. Nesse caso, como a linguagem avaliada é a própria linguagem do modelo G_n, a matriz de transição de estados é construída com todas as transições existentes no modelo, ou

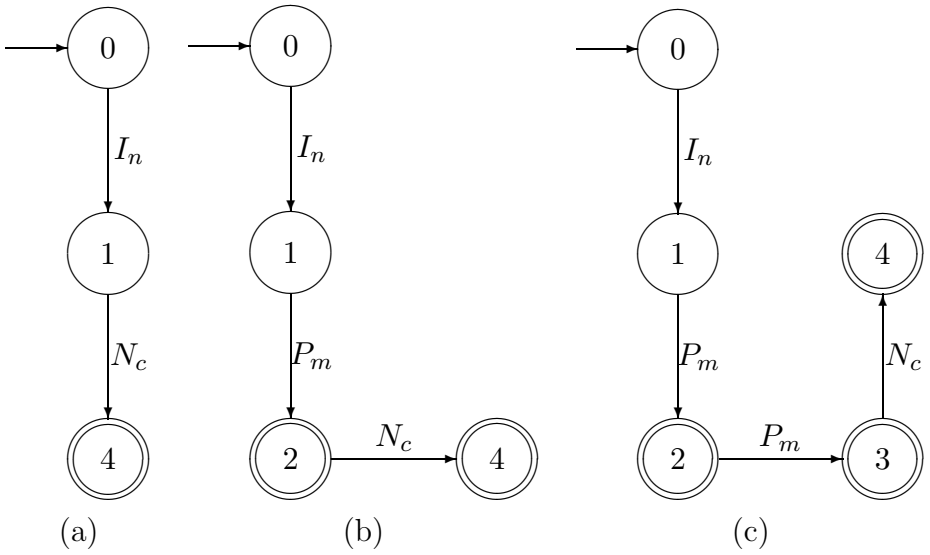


Figura 4.8: Diagrama de transição de estados do sistema controlado: (a) S_1/G_n; (b) S_2/G_n; (c) S_3/G_n.

seja:

$$\mathbf{\Pi} = \begin{bmatrix} 0 & 0,3 & 0 & 0 & 0 \\ 0 & 0 & 0,3 & 0 & 0,3 \\ 0 & 0 & 0 & 0,3 & 0,3 \\ 0 & 0 & 0 & 0 & 0,3 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}. \tag{4.61}$$

Com a matriz $\mathbf{\Pi}$ e o vetor característico, $\underline{\mathcal{X}}^T = \begin{bmatrix} 0 & 0 & -0,1 & -0,2 & 1 \end{bmatrix}$, o vetor $\boldsymbol{\mu}$ pode ser calculado através da aplicação direta da equação (4.60), ou seja:

$$\begin{aligned} \boldsymbol{\mu} &= (\mathbf{I} - \mathbf{\Pi})^{-1}\underline{\mathcal{X}} \\ \boldsymbol{\mu} &= \begin{bmatrix} 1 & -0,3 & 0 & 0 & 0 \\ 0 & 1 & -0,3 & 0 & -0,3 \\ 0 & 0 & 1 & -0,3 & -0,3 \\ 0 & 0 & 0 & 1 & -0,3 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}^{-1} \begin{bmatrix} 0 \\ 0 \\ -0,1 \\ -0,2 \\ 1 \end{bmatrix} = \begin{bmatrix} 0,1107 \\ 0,3690 \\ 0,2300 \\ 0,1000 \\ 1,0000 \end{bmatrix}. \end{aligned} \tag{4.62}$$

Uma vez obtido o vetor $\boldsymbol{\mu}$ para as diferentes sublinguagens de $\mathcal{L}(G)$ geradas por diferentes controladores, basta então comparar o primeiro elemento de cada vetor $\boldsymbol{\mu}$ obtido. O controlador que apresentar o melhor desempenho, segundo o critério de medida de linguagem, será o controlador cuja linguagem gerada a partir do estado inicial x_0, possuir o maior valor de medida de linguagem, ou seja, a linguagem controlada com o maior valor de μ_0.

Considere como exemplo três controladores (S_1, S_2 e S_3) e o autômato de na-

vegação do exemplo anterior, G_n, sendo os diagramas de transição de estados dos sistemas controlados apresentados na figura 4.5.2. Considerando apenas as transições de estados existentes em cada um dos modelos da figura 4.5.2, é possível construir as matrizes de transição de estados para cada um desses modelos. Para o sistema controlado S_1/G_n a matriz de transição de estados é:

$$\mathbf{\Pi}_{S_1} = \begin{bmatrix} 0 & 0,3 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0,3 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}. \tag{4.63}$$

Substituindo-se a matriz $\mathbf{\Pi}_{S_1}$ na equação (4.60), obtém-se o seguinte vetor de medida de linguagem:

$$\boldsymbol{\mu}_{S_1} = \begin{bmatrix} 0,0900 & 0,3000 & -0,100 & -0,200 & 1,000 \end{bmatrix}. \tag{4.64}$$

Para o sistema controlado S_2/G_n a matriz de transição de estados é:

$$\mathbf{\Pi}_{S_2} = \begin{bmatrix} 0 & 0,3 & 0 & 0 & 0 \\ 0 & 0 & 0,3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0,3 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \tag{4.65}$$

e o vetor de medida de linguagem é:

$$\boldsymbol{\mu}_{S_2} = \begin{bmatrix} 0,0180 & 0,0600 & 0,2000 & -0,200 & 1,000 \end{bmatrix}. \tag{4.66}$$

Para o sistema controlado S_3/G_n a matriz de transição de estados é:

$$\mathbf{\Pi}_{S_3} = \begin{bmatrix} 0 & 0,3 & 0 & 0 & 0 \\ 0 & 0 & 0,3 & 0 & 0 \\ 0 & 0 & 0 & 0,3 & 0 \\ 0 & 0 & 0 & 0 & 0,3 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \tag{4.67}$$

e o vetor de medida de linguagem é:

$$\boldsymbol{\mu}_{S_3} = \begin{bmatrix} -0,0063 & -0,0210 & -0,0700 & 0,1000 & 1,000 \end{bmatrix}. \tag{4.68}$$

Com as equações (4.64), (4.66) e (4.68), é possível comparar o desempenho das

linguagens controladas através do valor do elemento μ_0 de cada um dos vetores $\boldsymbol{\mu}_{S_i}$. Nesse exemplo, o sistema S_1/G_n apresenta o melhor desempenho ($\mu_0 = 0,0900$), com os dois motores funcionando normalmente durante toda a navegação. O sistema S_2/G_n representa o sistema quando um problema é detectado em um dos motores apenas e seu índice de desempenho é mais baixo que o anterior ($\mu_0 = 0,0180$). O pior índice de desempenho ($\mu_0 = -0,0063$) para o exemplo considerado é do modelo S_3/G_n, que representa o sistema de navegação quando um problema é detectado nos dois atuadores do robô.

Capítulo 5

Arquitetura de navegação proposta

Considerando o espectro contínuo de classificação apresentado por Ronald C. Arkin em [2], a arquitetura de navegação proposta nesse trabalho pode ser classificada como prioritariamente deliberativa. As teorias de sistemas a eventos discretos e de controle supervisório, apresentadas nos capítulos 3 e 4, serão utilizadas para a modelagem e composição de diversos comportamentos deliberativos e um prioritariamente reativo. A utilização de comportamentos independentes para compor essa arquitetura a qualifica como uma arquitetura baseada em comportamento. A estrutura geral da arquitetura de navegação proposta é apresentada na figura 5.1. Nessa figura, é possível observar a característica hierárquica dessa arquitetura, onde as informações providas pelo planejador devem ser executadas pelo controlador de baixo nível, o qual não tem qualquer ligação direta com o planejador.

Para o desenvolvimento deste trabalho, será considerada conhecida a planta baixa do ambiente que se deseja navegar. Esse ambiente será modelado utilizando a teoria de sistemas a eventos discretos e o modelo será utilizado para realizar o planejamento da trajetória a ser executada pelo robô (planejador na figura 5.1). Utilizando a mesma teoria, serão modelados os comportamentos admissíveis para o robô, necessários para executar a tarefa de navegar nesse ambiente (supervisor na figura 5.1). A composição destes comportamentos será responsável por passar as informações provenientes do planejador para serem executadas pelo robô. As informações sensoriais realimentam o supervisor indicando a finalização de uma tarefa ou um outro evento qualquer que tenha sido modelado nesse supervisor.

Esta capítulo está organizado da seguinte forma. Na seção 5.1 é apresentado o problema abordado neste trabalho, bem como o robô que será utilizado como plataforma de trabalho. Na seção 5.2, os autômatos que modelam o robô e os comportamentos admissíveis no ambiente de navegação são construídos e utilizados para compor um único modelo que representa todo o sistema. Na seção 5.3 é apresentada

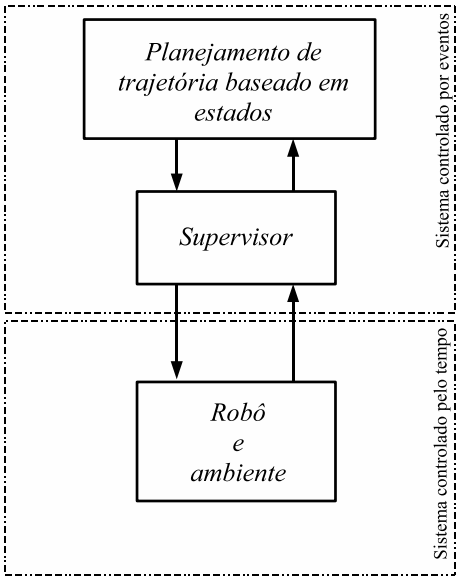


Figura 5.1: Diagrama de blocos da arquitetura de navegação proposta.

a construção do autômato de planejamento, bem como a sua utilização para a determinação da melhor trajetória a ser executada pelo robô. A seguir, na seção 5.4, são apresentados os detalhes de implementação em *software* do sistema proposto, bem como os resultados obtidos, em simulação, utilizando esse sistema.

5.1 Formulação do problema

A modelagem proposta nesse trabalho tem como característica principal a discretização das ações permitidas para o robô, ou seja, ao robô será permitido apenas movimentos específicos de translação e rotação desacoplados, por exemplo girar 90 graus ou transladar 1 metro. Para definir os movimentos necessários ao robô na tarefa de navegação, é preciso definir inicialmente o ambiente onde o robô irá navegar. O ambiente considerado nesse trabalho está representado na figura 5.2. Esse mapa hipotético representa um depósito onde a tarefa do robô é, partindo do ponto inicial, posicionar-se o mais próximo possível do ponto de destino, para a retirada de um item do depósito, e depois retornar ao ponto inicial onde este item será entregue, finalizando a tarefa. É importante salientar que o foco deste trabalho é realizar a navegação da plataforma móvel no ambiente utilizando a arquitetura proposta. Assim sendo, não serão abordadas as etapas da tarefa descrita acima referentes ao controle de manipuladores e retirada do item do depósito, por exemplo.

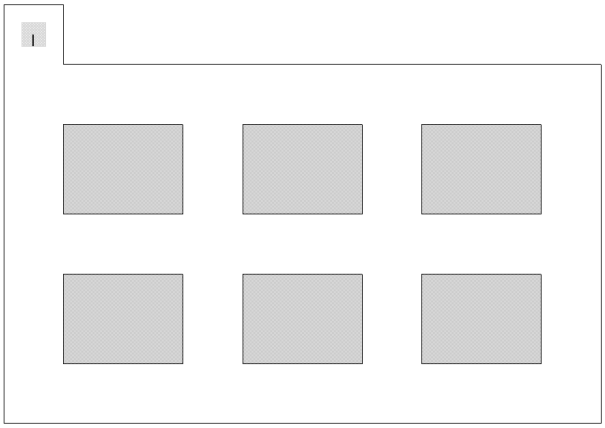


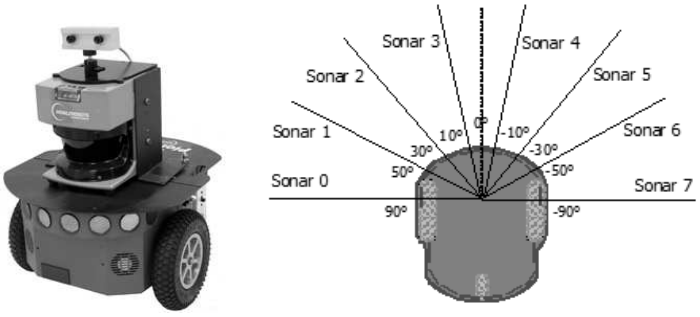
Figura 5.2: Ambiente de navegação considerado.

5.1.1 O robô P3-DX

A plataforma móvel utilizada para o desenvolvimento deste trabalho foi o robô móvel *Pioneer P3-DX* da *Mobile Robots Inc.*, ilustrado na figura 5.3(a). Este robô possui, em sua configuração mais simples, sensores de distância ultrassônicos dispostos na região frontal como ilustrado na figura 5.3(b) e *encoders* que permitem a localização incremental do robô no ambiente por intermédio da odometria. A tração do P3-DX é do tipo diferencial com dois atuadores independentes, um em cada roda. Esse robô foi escolhido por ser um dos mais utilizados pelos pesquisadores da área de robótica móvel, sendo a sua cinemática de fácil compreensão por se tratar de um robô a tração diferencial. Uma outra característica que levou à escolha desta plataforma é o *software* livre MobileSim, disponibilizado pela *Mobile Robots Inc.*. MobileSim é um simulador capaz de fazer o mesmo papel do robô em um ambiente, considerando a dinâmica e a cinemática do mesmo e comunicando-se com o controlador usando o mesmo protocolo de comunicação que o robô real utilizaria. Dessa forma, é possível desenvolver um sistema de navegação utilizando somente o simulador e, com nenhuma ou quase nenhuma alteração, utilizar este mesmo sistema em um robô real, o que possibilita o desenvolvimento de pesquisas na área de robótica móvel, antes mesmo da aquisição de um robô propriamente dito.

5.2 Autômatos de navegação - Supervisor

Para navegar no ambiente ilustrado na figura 5.2 um robô precisa apenas de movimentos desacoplados de translação e rotação, sendo que as rotações devem ocorrer apenas em alguns pontos de interesse (intersecção dos corredores). Esses pontos são chamados de *pontos de transição*. Analisando o mapa da figura 5.4, é possível observar 12 diferentes valores de distâncias de translação necessárias ao robô para



(a) (b)

Figura 5.3: (a) Plataforma móvel *Pioneer P3-DX* (*Mobile Robots Inc.*); (b) Disposição dos sensores ultrassônicos no robô P3-DX.

navegar neste ambiente, dos quais 8 são comandos de translação com valores fixos (que indicam as distâncias entre os pontos de transição) e os demais variáveis; estes últimos indicam ou a distância entre a posição atual do robô (R) e o último ponto de transição válido (T), ou a distância necessária para ir do ponto de transição atual (P) ao destino (D).

As distâncias ilustradas na figura 5.4 estão associadas a eventos de acordo com a tabela 5.1. Nas tabelas utilizadas nesse trabalho para descrever os eventos, o símbolo $\surd$ significa que o evento possui a propriedade e o símbolo $\times$ significa que o evento não possui a propriedade associada. Apesar de associar os nomes dos eventos às distâncias verticais e horizontais, separadamente, não existe diferença na implementação de um comando de translação vertical ou horizontal para o robô. Isso porque sempre que o robô receber uma ordem para executar uma translação, ele deverá estar orientado na direção do movimento, sendo necessário apenas percorrer a distância definida pelo evento, independentemente desta distância ser horizontal ou vertical. No que diz respeito à rotação, devido à simplicidade do mapa escolhido, é necessário apenas três valores de rotação para o robô, quais sejam: (*i*) 90 graus em sentido anti-horário; (*ii*) 90 graus em sentido horário e; (*iii*) 180 graus em qualquer sentido. Essas rotações estão associadas a eventos de acordo com a tabela 5.1. Todos esses eventos (tabela 5.1) são controláveis e observáveis, uma vez que, em sua essência, são comandos de ação dados ao robô para que este os execute. Para fins de simplificação, será utilizado nesse trabalho o conjunto de eventos

$$E = \{v_1, v_2, v_3, v_4, v_5, v_f, v_r, h_1, h_2, h_3, h_f, h_r, r_1, r_2, r_3\} \tag{5.1}$$

para denotar o conjunto de todos os eventos de movimento. Note que

$$E = E_r \dot{\cup} E_t, \tag{5.2}$$

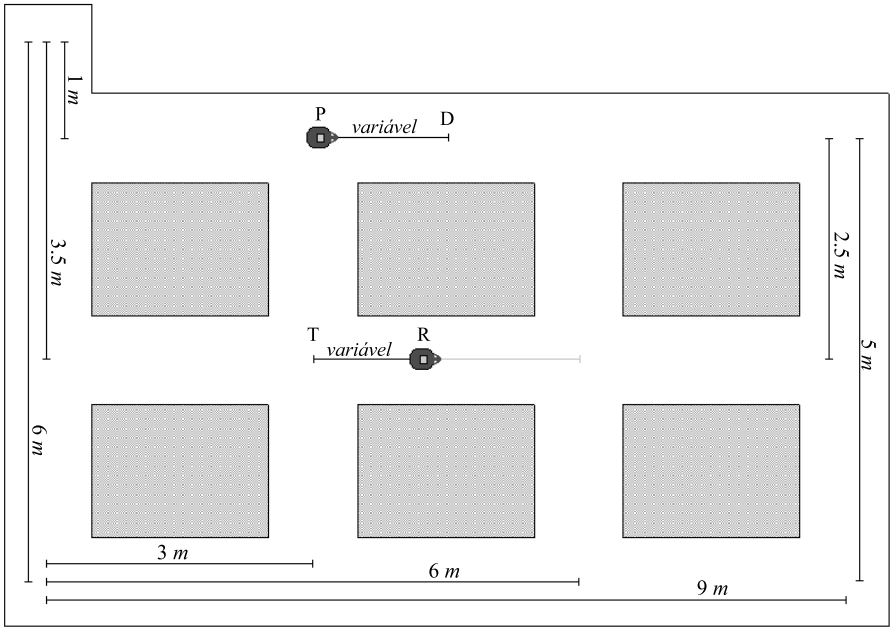


Figura 5.4: Distâncias de translação necessárias ao robô para navegar no ambiente considerado.

sendo

$$E_r = \{r_1, r_2, r_3\} \text{ e } E_t = \{v_1, v_2, v_3, v_4, v_5, v_f, v_r, h_1, h_2, h_3, h_f, h_r\}, \tag{5.3}$$

em que o conjunto E_t representa os eventos de translação e o conjunto E_r representa os eventos de rotação.

O próximo passo para a obtenção do autômato de navegação, é construir o modelo a eventos discretos G do robô e seus movimentos admissíveis no ambiente. O autômato de navegação G é obtido a partir da composição de três comportamentos mais simples (G_1, G_2 e G_3), quais sejam: (i) Translação e rotação desacoplados (G_1) ; (ii) Replanejar ao detectar um obstáculo (G_2) e; (iii) Interface operação-planejamento-execução (G_3). Os modelos dos comportamentos G_1, G_2 e G_3 são apresentados a seguir.

5.2.1 Modelagem dos comportamentos

Comportamento 1: Translação e rotação desacoplados

Para navegar no ambiente de navegação proposto, o robô precisa apenas de movimentos de translação e rotação desacoplados, ou seja, não há necessidade de movimentos em arco, por exemplo. Além disso, é necessário garantir que uma nova ordem de movimento só poderá ser dada ao robô caso o robô esteja disponível (não esteja executando algum outro movimento), do contrário, o comando de movimento

Tabela 5.1: Eventos que indicam os comandos de ação admissíveis do robô.

Evento	Controlável	Observável	Descrição
v_1	$\surd$	$\surd$	efetuar uma translação de 1 m (vertical)
v_2	$\surd$	$\surd$	efetuar uma translação de 2.5 m (vertical)
v_3	$\surd$	$\surd$	efetuar uma translação de 5 m (vertical)
v_4	$\surd$	$\surd$	efetuar uma translação de 3.5 m (vertical)
v_5	$\surd$	$\surd$	efetuar uma translação de 6 m (vertical)
v_f	$\surd$	$\surd$	efetuar uma translação para o destino (vertical)
v_r	$\surd$	$\surd$	efetuar uma translação de retorno (vertical)
h_1	$\surd$	$\surd$	efetuar uma translação de 3 m (horizontal)
h_2	$\surd$	$\surd$	efetuar uma translação de 6 m (horizontal)
h_3	$\surd$	$\surd$	efetuar uma translação de 9 m (horizontal)
h_f	$\surd$	$\surd$	efetuar uma translação para o destino (horizontal)
h_r	$\surd$	$\surd$	efetuar uma translação de retorno (horizontal)
r_1	$\surd$	$\surd$	efetuar uma rotação de 90 graus (anti-horário)
r_2	$\surd$	$\surd$	efetuar uma rotação de 90 graus (horário)
r_3	$\surd$	$\surd$	efetuar uma rotação de 180 graus

deve ser desprezado.

Para garantir que uma nova ordem seja dada ao robô apenas quando este não estiver executando uma outra ação, é necessário utilizar dois eventos: (i) w_0, que indica que a velocidade de rotação (velocidade angular) do robô é zero, ou seja, o robô não está realizando rotação e; (ii) v_0, que indica que a velocidade de translação (velocidade linear) do robô é zero, ou seja, o robô não está realizando translação. Esses eventos não controláveis (sua ocorrência não pode ser desabilitada por um supervisor) são apresentados na tabela 5.2, juntamente com o evento tr que indica que o robô terminou a navegação, ou seja, alcançou o ponto de destino. Para fins de simplificação, será utilizado nesse trabalho o conjunto de eventos:

$$E_f = \{v_0, w_0, tr\} \tag{5.4}$$

para denotar o conjunto de todos os eventos que indicam o término de um movimento (rotação ou translação) ou da tarefa como um todo.

O autômato G_1, que modela esse comportamento, é apresentado na figura 5.5. Observando o modelo proposto, é possível notar que, uma vez que o robô comece a executar uma ação de rotação (ou translação), ele só irá aceitar um novo comando quando terminar de executar essa tarefa, ou seja, quando a velocidade angular (ou linear) for zero, o que é o objetivo desse comportamento. Dessa forma os movimentos de translação e rotação jamais poderão ocorrer ao mesmo tempo. Além disso, existe um *self loop* no estado 0 com o evento tr. Essa transição indica que o robô chegou

Tabela 5.2: Eventos que indicam o término de uma tarefa de translação, rotação ou da navegação.

Evento	Controlável	Observável	Descrição
v_0	×	$\surd$	indica velocidade linear nula
w_0	×	$\surd$	indica velocidade angular nula
tr	×	$\surd$	tarefa realizada com sucesso

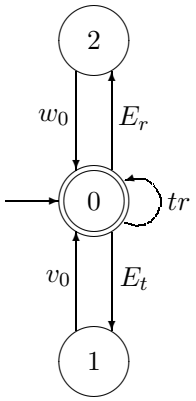


Figura 5.5: Autômato G_1 que modela o comportamento 1.

ao seu destino e, por isso, só pode ocorrer quando o robô estiver livre (não estiver executando um movimento), ou seja, no estado 0. Note também que o estado inicial (estado 0) é marcado, pois indica a realização de um movimento com sucesso, seja este movimento de translação, rotação ou mesmo da navegação como um todo (chegar ao ponto de destino).

Comportamento 2: Replanejar ao detectar um obstáculo

Esse comportamento tem por objetivo modelar a capacidade de reação do robô ao encontrar um obstáculo, possibilitando assim uma navegação mais segura. Mesmo em um sistema de navegação prioritariamente deliberativo, é necessário incluir na arquitetura de navegação a capacidade de evitar colisões, fazendo com que o robô, ao observar um obstáculo em seu caminho, possa interromper a sua ação e replanejar a execução da tarefa. Para modelar esse comportamento é necessário fazer com que, sempre que o robô detectar a presença de um obstáculo em seu caminho, ele execute uma sequência de eventos capaz de levá-lo ao ultimo ponto de transição válido, onde a execução da tarefa será replanejada. Para realizar a modelagem desse comportamento, é necessária a criação de alguns eventos definidos na tabela 5.3. Para fins de simplificação, será utilizado nesse trabalho o conjunto de eventos:

$$E_o = \{od, pr, rp, rpe\} \tag{5.5}$$

Tabela 5.3: Eventos relacionados à detecção de obstáculos e replanejamento.

Evento	Controlável	Observável	Descrição
od	$\surd$	$\surd$	obstáculo detectado pelo robô
pr	$\surd$	$\surd$	ordem para parar o robô
rp	$\surd$	$\surd$	ordem para replanejar a execução da tarefa
rpe	$\times$	$\surd$	replanejamento executado

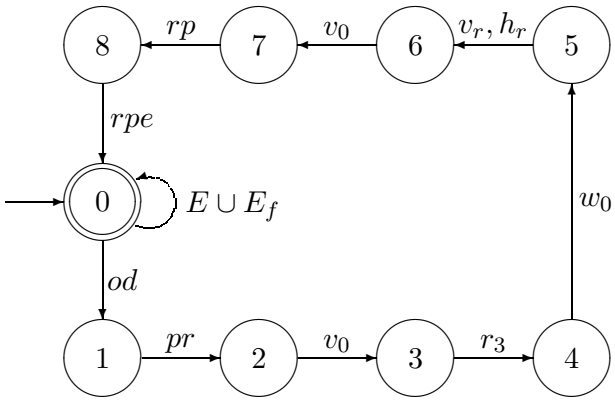


Figura 5.6: Autômato G_2 que modela o comportamento 2.

para denotar o conjunto de todos os eventos relacionados com a detecção de obstáculo. Com esses eventos, é possível construir um autômato G_2 que modela esse comportamento. O diagrama de transição de estados do autômato G_2 desenvolvido está representado na figura 5.6.

Observando o modelo proposto, é possível notar que, enquanto nenhum obstáculo é detectado, todos os eventos de movimento (eventos em E) e os eventos que indicam realização de uma tarefa (eventos em E_f) podem ocorrer normalmente (estado 0). Uma vez que o robô detecte um obstáculo (evento od), o único evento permitido é o comando de parar o robô (evento pr). Quando o robô atender a este comando será gerado o evento v_0, que indica que o robô parou. Em seguida ele irá retornar ao último ponto de transição válido (através da sequência de eventos $r_3 w_0 v_r v_0$ ou $r_3 w_0 h_r v_0$, dependendo da orientação do robô), onde irá replanejar a execução da tarefa (evento rp) por um caminho diferente. Assim que o replanejamento for concluído (evento rpe), o robô volta ao seu funcionamento normal, podendo novamente executar a sequência de navegação planejada, atingindo, assim, o objetivo desse comportamento.

Note que o estado inicial (estado 0) é marcado, uma vez que esse estado indica que nenhum obstáculo foi detectado à frente do robô, ou seja, o robô pode navegar em segurança. Note também que o evento od, apesar de ser uma ação não previsível, é um evento controlável. Isso porque a detecção de um obstáculo pelo robô depende

Tabela 5.4: Eventos relacionados à interface operação-planejamento-execução.

Evento	Controlável	Observável	Descrição
p	$\surd$	$\surd$	ordem para planejar a execução da tarefa
pe	$\times$	$\surd$	planejamento executado
an	$\surd$	$\surd$	ordem para ativar a navegação
dn	$\surd$	$\surd$	ordem para desativar a navegação

da leitura dos sensores de distância que, por sua vez, podem ser habilitados ou desabilitados pelo supervisor quando for necessário, o que caracteriza a controlabilidade do evento.

Comportamento 3: Interface operação-planejamento-execução

Os comportamentos modelados em G_1 e G_2 representam apenas as possíveis ações que o robô tem permissão para executar com o intuito de atingir um ponto de destino no ambiente. No entanto, a definição desse ponto de destino e a sequência de eventos que o robô deve executar para alcançá-la não aparecem nesse modelo. O comportamento 3 tem como objetivo modelar a sequência de etapas necessárias ao robô, antes que este inicie a navegação propriamente dita, e estabelecer a relação entre o operador (pessoa responsável por informar o ponto de destino ao robô e pela ativação/desativação da navegação) e o sistema.

O diagrama de transição de estados do autômato G_3, que modela esse comportamento, é apresentado na figura 5.7 e os eventos associados a esse modelo são apresentados na tabela 5.4. No estado inicial (estado 0 do autômato G_3), o único evento permitido é a ordem para iniciar o planejamento (evento p). Esse evento é uma ação controlada pelo operador, a partir da qual o ponto de destino é informado ao sistema de planejamento de trajetória. No estado 1 o sistema está planejando a sequência de eventos que o robô deve executar para atingir o ponto de destino estipulado. Quando o planejamento for finalizado, ocorrerá o evento não controlável pe, que levará o autômato G_3 para o estado 2. Nesse estado, o robô está pronto para iniciar a navegação (execução da sequência de eventos planejada), que pode ser habilitada ou desabilitada, a qualquer momento, pelo operador por intermédio dos eventos controláveis an (ativar a navegação) e dn (desativar a navegação). Quando a navegação é ativada, o sistema passa para o estado 3 (robô navegando), onde permanece até que o operador interrompa a navegação (através do evento dn) ou que ocorra o evento tr, que indica que o robô completou a sequência planejada, ou seja, alcançou o ponto de destino. Uma vez ocorrido o evento tr, o autômato volta para o estado inicial para aguardar nova instrução de planejamento e reiniciar o processo de navegação. O estado 3 é marcado para indicar o estado em que a navegação está

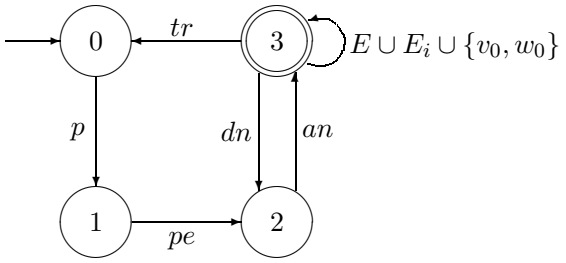


Figura 5.7: Autômato G_3 que modela o comportamento 3.

ativa.

5.2.2 Autômato de navegação não-controlado

O autômato de navegação não controlado G é obtido através da composição paralela dos três comportamentos (G_1, G_2 e G_3) que modelam a navegação do robô, ou seja:

$$G = G_1 \| G_2 \| G_3. \tag{5.6}$$

O diagrama de transição de estados do autômato G está representado na figura 5.8. Note que os estados foram renomeados com o intuito de simplificar a notação. Analisando esse autômato é possível observar que existem sequências de eventos indesejadas no sistema, como aquelas que terminam na detecção de um obstáculo enquanto o robô está parado (evento *od* do estado 3 para o estado 24), por exemplo. Uma das maneiras de se corrigir esse problema é utilizando a teoria de controle supervisório que, a partir de especificações associadas ao modelo do robô, produzirá um sistema controlado, cuja linguagem representa exatamente o comportamento desejado para o sistema de navegação, eliminando, assim, as sequências indesejadas. A especificação desenvolvida neste trabalho é apresentada na seção 5.2.3 e o sistema de navegação controlado, obtido a partir da composição dessa especificação e do autômato G, é apresentado na seção 5.2.4.

5.2.3 Modelagem da especificação de controle

No modelo do sistema de navegação G, a capacidade de detectar obstáculos está ativa independente da ação que o robô esteja executando. Isso é desnecessário, uma vez que o robô, parado ou executando uma rotação, jamais irá detectar um obstáculo estático. Esse fato não representa problema algum para o comportamento do sistema de navegação G, mas aumenta a complexidade desse modelo significativamente. Para fazer com que a capacidade de detectar obstáculos do robô esteja ativa somente quando este estiver realizando uma translação, é necessário criar uma especificação

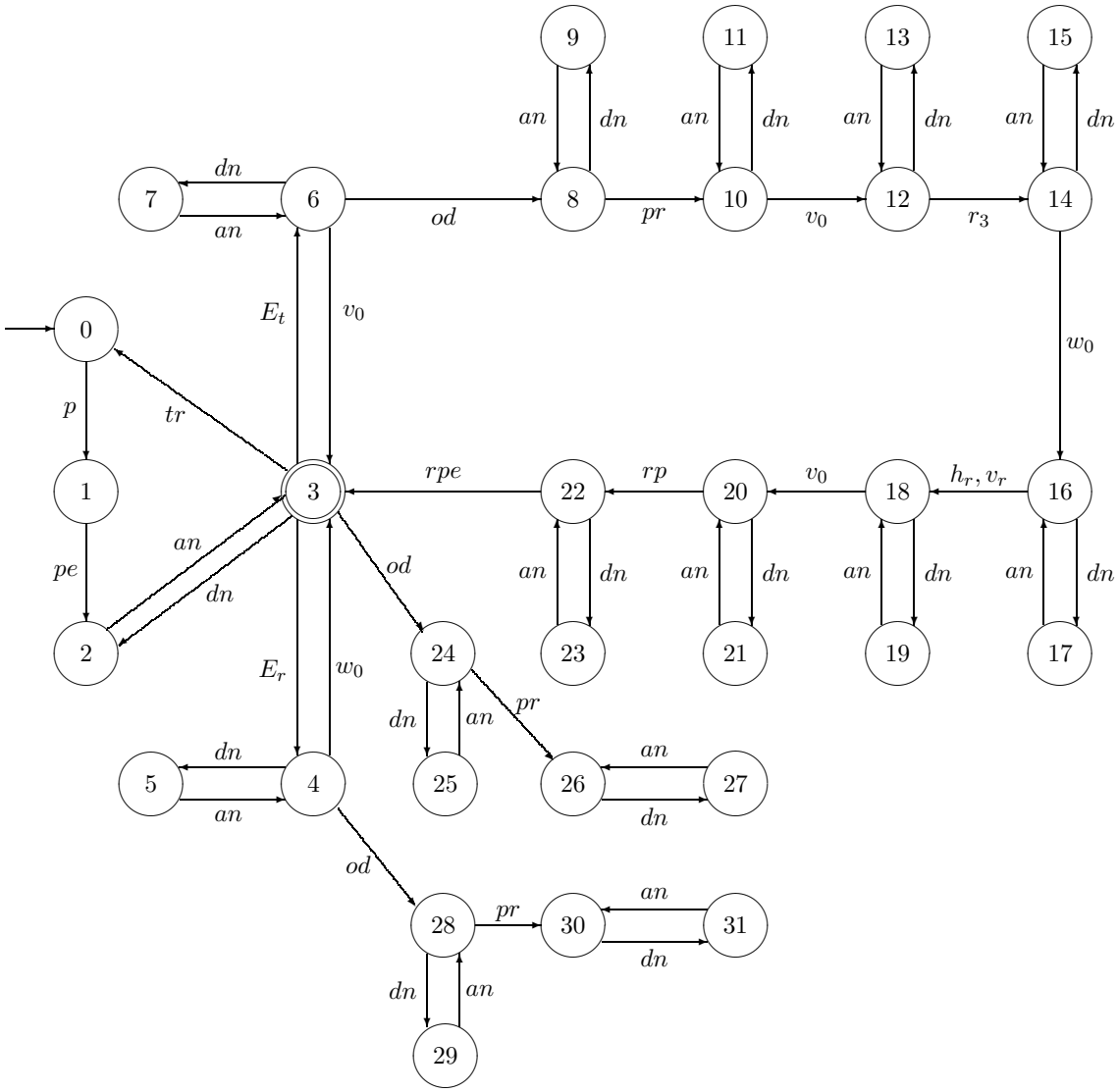


Figura 5.8: Autômato de navegação G não-controlado.

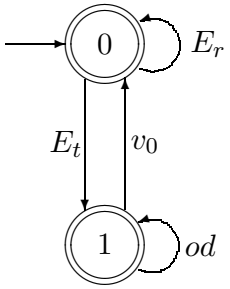


Figura 5.9: Autômato E_1 que modela a especificação 1.

de controle. O autômato E_1, representado na figura 5.9, capta de forma simples o comportamento desejado para essa especificação. No estado 0, onde os eventos de rotação (eventos em E_r) podem ocorrer, a função de detecção de obstáculos está desabilitada (*od* não pode ocorrer). Quando acontece qualquer evento de translação (eventos em E_t), o autômato passa para o estado 1, onde a detecção de obstáculos é habilitada (*od* pode ocorrer), até que ocorra o evento v_0, indicando que o movimento de translação foi finalizado, e o modelo volta para o estado inicial, reiniciando o processo. Os estados dessa especificação são marcados para não alterar a marcação do sistema como um todo, já que seus estados, individualmente, não representam o término de uma tarefa ou alguma outra informação que mereça destaque.

5.2.4 Autômato de navegação controlado

Uma vez definida a planta G e sua especificação de controle E_1, é preciso construir o autômato que represente a especificação de controle e o comportamento do robô em um único modelo. Esse modelo pode ser obtido realizando-se a composição paralela entre o autômato que modela o robô e a especificação de controle, conforme foi apresentado na seção 3.4.2.

Para um sistema supervisório ser realizável, como explicado na seção 4.4, é preciso que as linguagens geradas pelo modelo da planta e das especificações sejam regulares. Uma linguagem é regular quando ela pode ser marcada por um autômato de estados finitos. Uma vez que a planta e a especificação desenvolvidas neste trabalho podem ser modeladas através de autômatos de estados finitos, a verificação da regularidade das linguagens é trivial, bastando para isso marcar todos os estados dos autômatos que modelam cada uma das linguagens em questão.

O autômato S/G, que modela o comportamento de navegação controlado, pode ser obtido através da composição paralela entre o autômato da planta G e da especificação E_1, ou seja:

$$\mathcal{L}(S/G) = \mathcal{L}(G \| E_1) \tag{5.7}$$

$$\mathcal{L}_m(S/G) = \mathcal{L}_m(G \| E_1). \tag{5.8}$$

O autômato S/G que modela o sistema de navegação controlado é apresentado na figura 5.10.

Para garantir que o autômato S/G está correto, é preciso garantir que a linguagem modelada pela especificação E_1 é controlável. É sabido que a linguagem $\mathcal{L}(E_1)$ será controlável, em relação à linguagem $\mathcal{L}(G)$ e ao conjunto de eventos não controláveis de G, $E_{uc} = \{v_0, w_0, tr, rpe, pe\}$, se a seguinte relação de inclusão for verdadeira:

$$\overline{\mathcal{L}(E_1)}E_{uc} \cap \mathcal{L}(G) \subseteq \overline{\mathcal{L}(E_1)}, \tag{5.9}$$

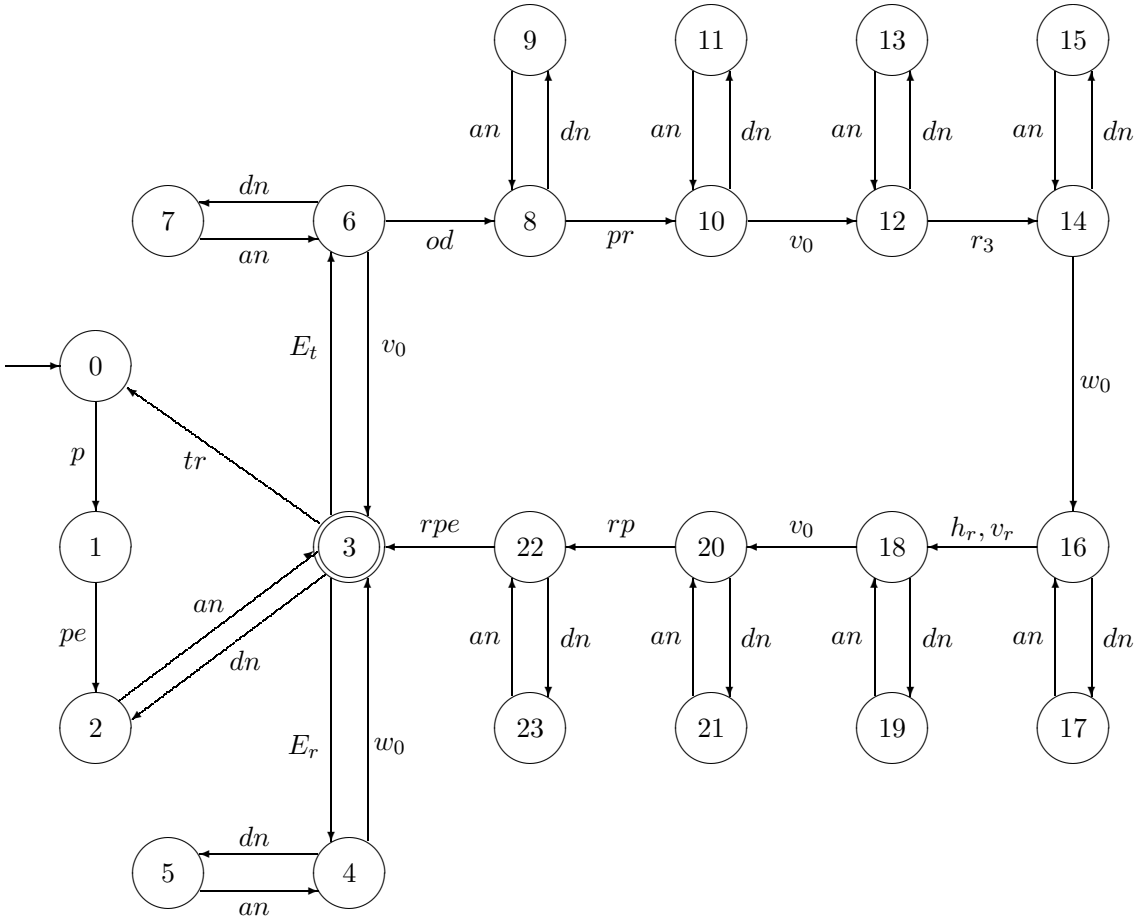


Figura 5.10: Autômato S/G que modela o comportamento do sistema supervisionado.

em que

$$\mathcal{L}(E_1) = \overline{\mathcal{L}(E_1)}, \tag{5.10}$$

ou seja, toda continuação não controlável da linguagem $\mathcal{L}(E_1)$ presente na linguagem $\mathcal{L}(G)$ deve também pertencer a $\mathcal{L}(E_1)$. Essa propriedade pode ser verificada comparando os estados do autômato G com os estados do autômato do sistema controlado S/G (teste de controlabilidade descrito no capítulo 4). Uma vez que todo evento não controlável em $\mathcal{L}(G)$ também aparece em $\mathcal{L}(E_1)$, a relação apresentada na equação (5.9) é verdadeira, logo, a linguagem $\mathcal{L}(E_1)$ é controlável.

Com o modelo do autômato S/G, o robô é capaz de executar a sequência de eventos definida pelo planejador (figura 5.1) de forma segura, realizando movimentos de translação, movimentos de rotação e solicitando um replanejamento da trajetória sempre que encontrar um obstáculo durante uma translação. Além disso, são permitidas intervenções do operador, habilitando ou desabilitando a navegação, sempre que julgar necessário.

5.3 Planejamento da trajetória baseado em estados - Planejador

O modelo do sistema supervisionado (figura 5.10) desenvolvido na seção 5.2, é responsável pela navegação segura do robô, fazendo com que este possa executar as sequências de eventos determinadas a partir do planejamento sempre que possível e solicitando um replanejamento quando necessário. Sendo assim, resta apenas definir como o planejamento da trajetória será executado. O planejamento de trajetória proposto leva em consideração os eventos de movimento admissíveis do robô (eventos em E_r e E_t) e o mapa do ambiente de navegação (figura 5.2) para possibilitar a construção de um autômato que modele todas as possíveis trajetórias que podem ser executadas pelo robô nesse ambiente. Esse autômato é denominado *autômato de planejamento*. Um vez construído corretamente o autômato de planejamento, a estratégia utilizada para escolher a melhor trajetória (sequência de eventos) consiste em separar as trajetórias que levam o robô do ponto atual ao ponto de destino e, em seguida, escolher a melhor trajetória, segundo o *critério de planejamento* μ.

5.3.1 Autômato de planejamento P

O autômato de planejamento P é um autômato que modela todas as possíveis trajetórias que podem ser executadas pelo robô no ambiente proposto. Considerando o mapa proposto como um plano cartesiano definido pelos eixos x e y, cada estado desse autômato representa uma tripla (x_r, y_r, θ_r), conforme ilustrado na figura 5.11. A variável x_r representa a posição x do robô no mapa, y_r representa a posição y do robô no mapa e θ_r representa o ângulo da orientação do robô no mapa, em relação ao eixo x.

Modelar os estados do autômato de planejamento P com todos os possíveis valores da tripla (x_r, y_r, θ_r), significaria um autômato com espaço de estados infinito, o que seria inviável considerando a abordagem proposta utilizando SED. Dessa forma, foram modelados apenas os estados onde o robô se encontra em um dos pontos de transição, definidos na seção 5.2. Todos os pontos de transição existentes no mapa do ambiente proposto aparecem representados por um "×" na figura 5.12. Para simplificar a notação da orientação e posição do robô no mapa, será utilizada uma representação equivalente para o robô, ilustrada na figura 5.13(a), ou seja, um ponto representando as coordenadas do robô no mapa (x_r e y_r) e uma seta representando a orientação do robô (θ_r). Considerando que o robô terá apenas orientações múltiplas de 90 graus (eventos r_1, r_2 e r_3), podem existir até 4 estados distintos em cada ponto de transição, a depender da orientação do robô. Os possíveis estados existentes em um mesmo ponto de transição são mostrados na figura 5.13(b) e todos os

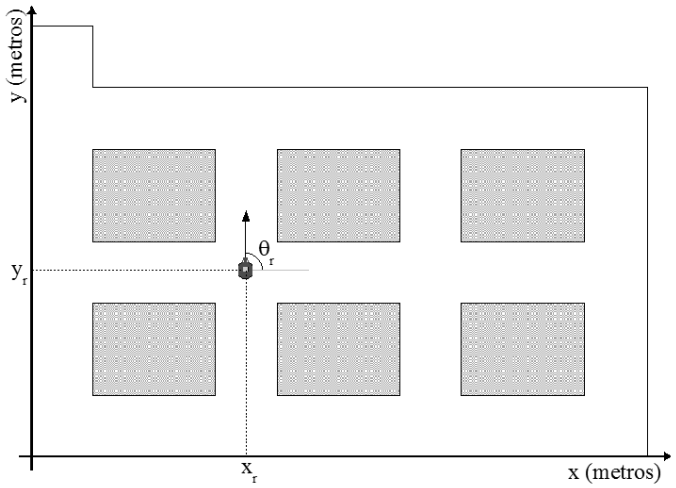


Figura 5.11: Representação das variáveis x_r, y_r e θ_r no mapa do ambiente.

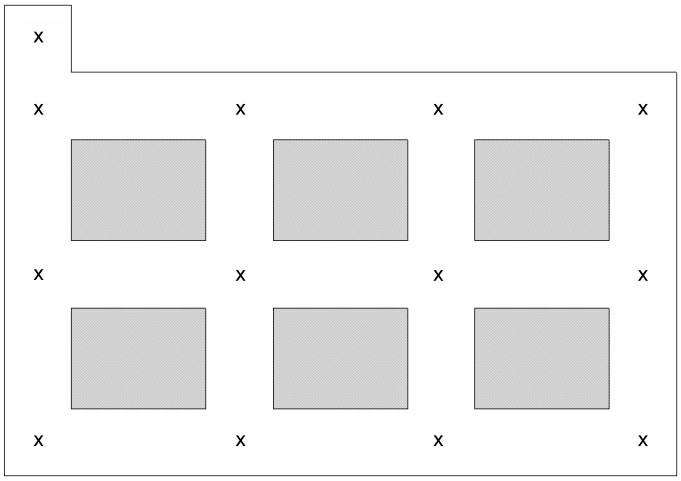


Figura 5.12: Representação de todos os pontos de transição existentes no mapa do ambiente.

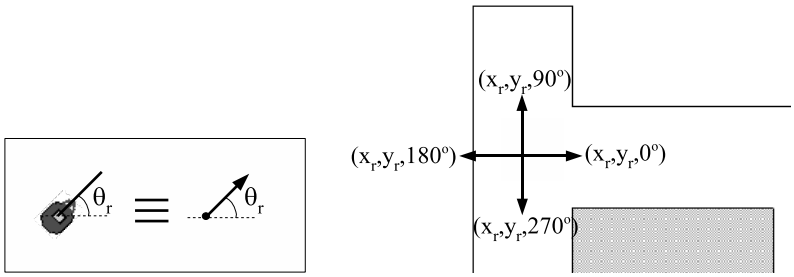


Figura 5.13: (a) Notação utilizada para simplificar a representação do robô no mapa; (b) Representação dos quatro estados admissíveis em um mesmo ponto de transição.

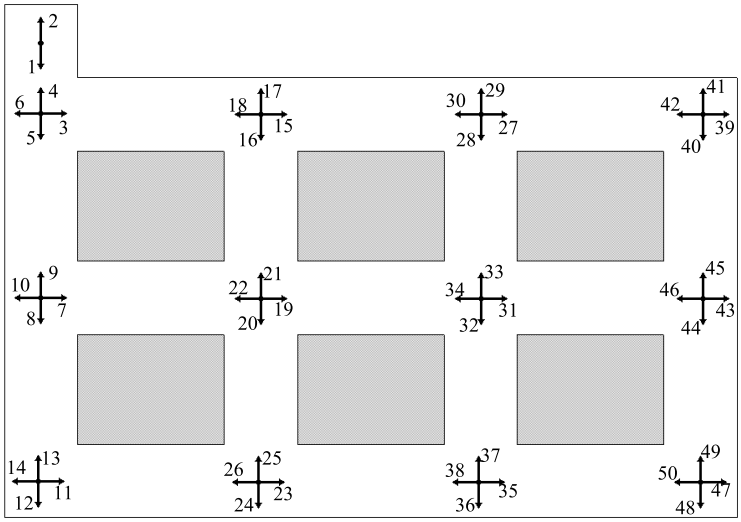


Figura 5.14: Representação e identificação no mapa dos estados associados a pontos de transição.

possíveis estados existentes em todos os pontos de transição do ambiente proposto estão representados na figura 5.14, utilizando a representação do robô, descrita na figura 5.13(a). A cada estado foi associado um número que identifica esse estado no autômato de planejamento, esses números podem ser observados na figura 5.14.

Uma vez definidos os estados do autômato P associados aos pontos de transição, é preciso definir, então, as transições existentes entre esses estados e associar os eventos de movimento do robô (eventos em E_t e E_r) a essas transições. Analisando o robô na sua posição inicial (estado 1), a ocorrência do evento v_1 (translação de 1 m na vertical) levaria o robô para o estado 5. Ainda no estado inicial, a ocorrência do evento v_4 (translação de 4.5 m na vertical) levaria o robô para o estado 8 ou a ocorrência do evento v_5 (translação de 6 m na vertical) levaria o robô para o estado 12. Essas transições de estado são apresentadas na figura 5.15(a). Analisando agora os possíveis movimentos do robô, quando este se encontra no estado 5, é possível definir transições associadas aos eventos v_2 (que leva ao estado 8), v_3 (que leva

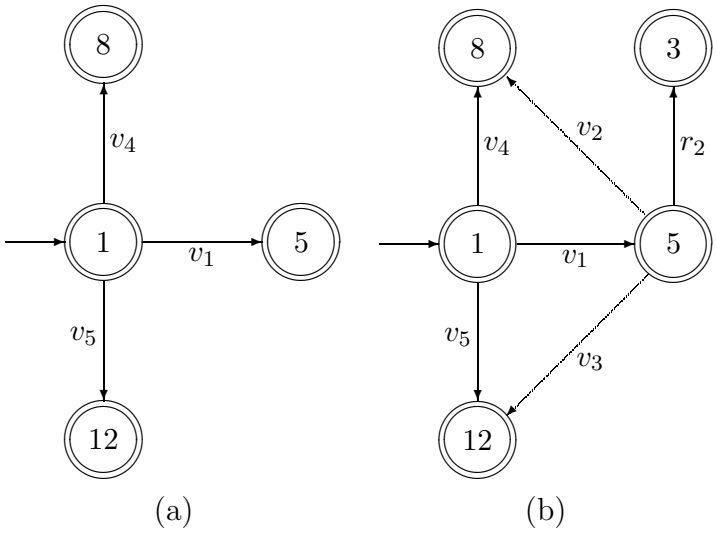


Figura 5.15: Construção do autômato de planejamento: (a) Eventos que partem do estado 1; (b) Eventos que partem dos estados 1 e 5.

ao estado 12) e r_2 (que leva ao estado 3). Note, também, que outras transições poderiam ser definidas no estado 5 como, por exemplo, a transição r_1 que levaria o robô para o estado 6 ou a transição r_3, que levaria o robô para o estado 4). No entanto, esses eventos não são considerados com o intuito de simplificar o autômato de planejamento, pois não existe razão para o robô realizar uma rotação que o deixaria orientado de frente para a parede ou que o orientasse para a direção de onde ele veio. Juntando as transições que partem do estado 1 e as transições que partem do estado 5 em um único autômato obtém-se o modelo ilustrado na figura 5.15(b). Repetindo o procedimento descrito para os estados 1 e 5, com todos os estados ilustrados na figura 5.14 obtém-se o autômato que modela todas as possíveis transições (associadas a eventos em E) entre estes estados. Com esse autômato é possível representar uma trajetória que leve o robô de um estado qualquer x_i a outro estado qualquer x_f, como sendo uma sequência de eventos s_p.

Com o modelo do autômato de planejamento atual é possível definir diferentes sequências de eventos que levam o robô de um estado a outro. No entanto, até o momento, foram incluídos apenas estados associados a pontos de transição. Para completar o autômato P, é preciso incluir um segundo grupo de estados nesse modelo, os *estados finais*. Os estados finais representam a posição de destino do robô, ou seja, a posição que o robô deve alcançar para completar a tarefa planejada. Em cada um dos corredores do ambiente existem dois estados finais, um para cada uma das orientações possíveis, conforme ilustrado na figura 5.16. Só é possível alcançar um dado estado final a partir de um único outro estado, por exemplo, para alcançar o estado final 59 é preciso chegar antes no estado 3 para, só então, ir para o estado 59; para alcançar o estado final 85 é preciso chegar antes ao estado 34, e assim por

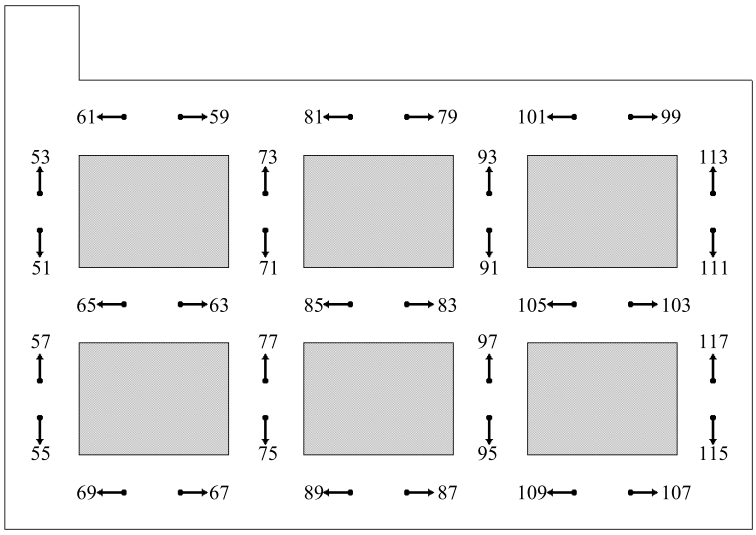


Figura 5.16: Representação e identificação dos estados finais no mapa do ambiente.

diante. No total, o autômato de planejamento P possui 118 estados, 15 eventos (eventos de movimento apresentados na tabela 5.1) e 219 transições.

A posição de um estado final no mapa do ambiente é variável, por isso, os eventos que levam o robô, do ponto de transição mais próximo, a um estado final são os eventos v_f (caso o movimento seja na vertical) ou h_f (caso o movimento seja na horizontal). Esses eventos, conforme descrito na seção 5.2, representam um valor de translação variável, que será definido automaticamente pelo sistema de planejamento assim que o operador definir o ponto de destino do robô e der a ordem para iniciar o planejamento (evento p do autômato do sistema supervisor S/G). Por exemplo, a trajetória ilustrada na figura 5.17 é definida pela sequência de eventos:

$$s_p = v_1 r_1 h_1 r_2 v_f. \tag{5.11}$$

Para a sequência de eventos s_p, gerada pelo planejamento no exemplo da figura 5.17, o valor associado ao evento v_f é uma translação na vertical de 1.7 metros.

5.3.2 Possíveis trajetórias

Utilizando o autômato de planejamento descrito na seção 5.3.1, é possível definir diversas sequências de eventos que levam o robô de um estado para outro. No entanto, dependendo do estado de destino, podem existir infinitas sequências que representem essa tarefa. Isso ocorre porque a estrutura do ambiente permite ao robô revisitar pontos de transição sem antes concluir a tarefa, criando assim laços de repetição infinitos. Para solucionar esse problema, o algoritmo de planejamento utilizado faz uma busca pelas possíveis trajetórias, criando uma árvore com todas

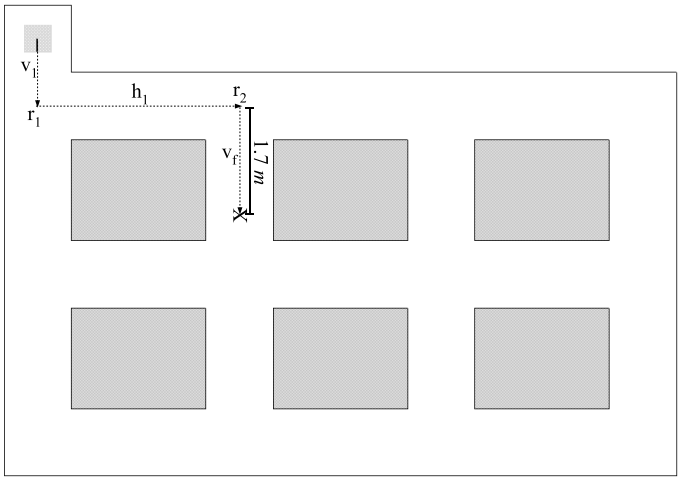


Figura 5.17: Exemplo de uma sequência de eventos planejada, s_p, onde o "X" representa o ponto de destino.

as sequências desde o estado atual do robô até o seu estado de destino. Essa busca possui quatro critérios de parada que indicam o fim de um dos ramos da árvore de busca, quais sejam:

1. CP1: A trajetória chegou a um estado pertencente a um ponto de transição pelo qual ele já passou anteriormente;

2. CP2: A trajetória chegou a um *estado final* (figura 5.16) diferente daquele associado ao ponto de destino do robô;

3. CP3: A trajetória contém duas rotações seguidas;

4. CP4: A trajetória chegou ao *estado final* associado ao ponto de destino do robô.

Os ramos da árvore que forem interrompidos com os critérios de parada CP1, CP2 e CP3 serão excluídos da busca, enquanto que os ramos que terminarem com o critério CP4 serão as possíveis trajetórias selecionadas. O algoritmo de busca baseado na construção de uma árvore, é apresentado a seguir.

Algoritmo 5.1 *:*

Passo 1 *Defina como a raiz da árvore o estado atual do robô no autômato P.*

Passo 2 *Crie os descendentes da raiz da seguinte forma:*
Suponha que $|\Gamma(x_i) \setminus \{v_3, v_4, v_5, h_2, h_3\}| = n_i$. Crie, então, n_i descendentes de x_i e rotule-os como $x' = f(x_i, \sigma)$, $\sigma \in \Gamma(x_i) \setminus \{v_3, v_4, v_5, h_2, h_3\}$. Atribua aos nós criados o status de ativo e à raiz o status de inativo.

Passo 3 *Para cada um dos nós ativos x_a da árvore faça:*

(i) *se pelo menos uma das condições CP1, CP2 ou CP3 for verdadeira, elimine o nó x_a e atribua a ele o status de inativo.*

(ii) *se a condição CP4 for verdadeira, então, o nó x_a é uma folha. Atribua a ele o status de inativo.*

(iii) *se nenhuma das condições CP1, CP2, CP3 ou CP4 for verdadeira, então, crie os descendentes do nó ativo x_a da seguinte forma:*
Suponha que $|\Gamma(x_a) \setminus \{v_3, v_4, v_5, h_2, h_3\}| = n_a$. Crie, então, n_a descendentes de x_a e rotule-os como $x' = f(x_a, \sigma)$, $\sigma \in \Gamma(x_a) \setminus \{v_3, v_4, v_5, h_2, h_3\}$. Atribua aos nós criados o status de ativo e ao nó x_a o status de inativo.

Passo 4 *Se existir algum nó ativo na árvore volte para o Passo 3. Caso contrário, vá para o Passo 5.*

Passo 5 *As possíveis trajetórias são as menores sequências de eventos que partem da raiz até as folhas. O número de possíveis trajetórias é igual ao número de folhas.*

Para exemplificar o algoritmo 5.1, considere, por exemplo, o ponto de destino marcado com um "X" na figura 5.17. Para esse ponto de destino, partindo do estado inicial 1, e utilizando o algoritmo de busca apresentado, ao invés de infinitas sequências, são obtidas 31 sequências como possíveis trajetórias para se atingir o destino. Um parte da árvore de busca gerada é apresentada na figura 5.18, na qual os ramos interrompidos sem alcançar o destino (através dos critérios CP1, CP2 e CP3) são marcados com um "$\times$" e os ramos que alcançaram o destino (interrompidos pelo critério de parada CP4 - folhas) são marcados com uma estrela. As possíveis trajetórias obtidas com essa busca são apresentadas na tabela 5.5 e denotadas por s_j, com $j = 1, 2, ..., 31$, para referências futuras. Note que na árvore de busca apresentada na figura 5.18, são definidos apenas os estados que o robô deverá passar em cada uma das possíveis trajetórias. A sequência de eventos associada a cada uma das trajetórias só é definida quando a trajetória estiver completa, buscando sempre a menor sequência de eventos que a represente. Por exemplo, seja uma trajetória selecionada na qual robô precisará sair do estado 5 e ir para o estado 12, então, a este movimento associa-se o evento v_3, que é a sequência de menor comprimento que o representa, desprezando, assim, a sequência v_2v_2, que também representa o mesmo movimento, mas com dois eventos.

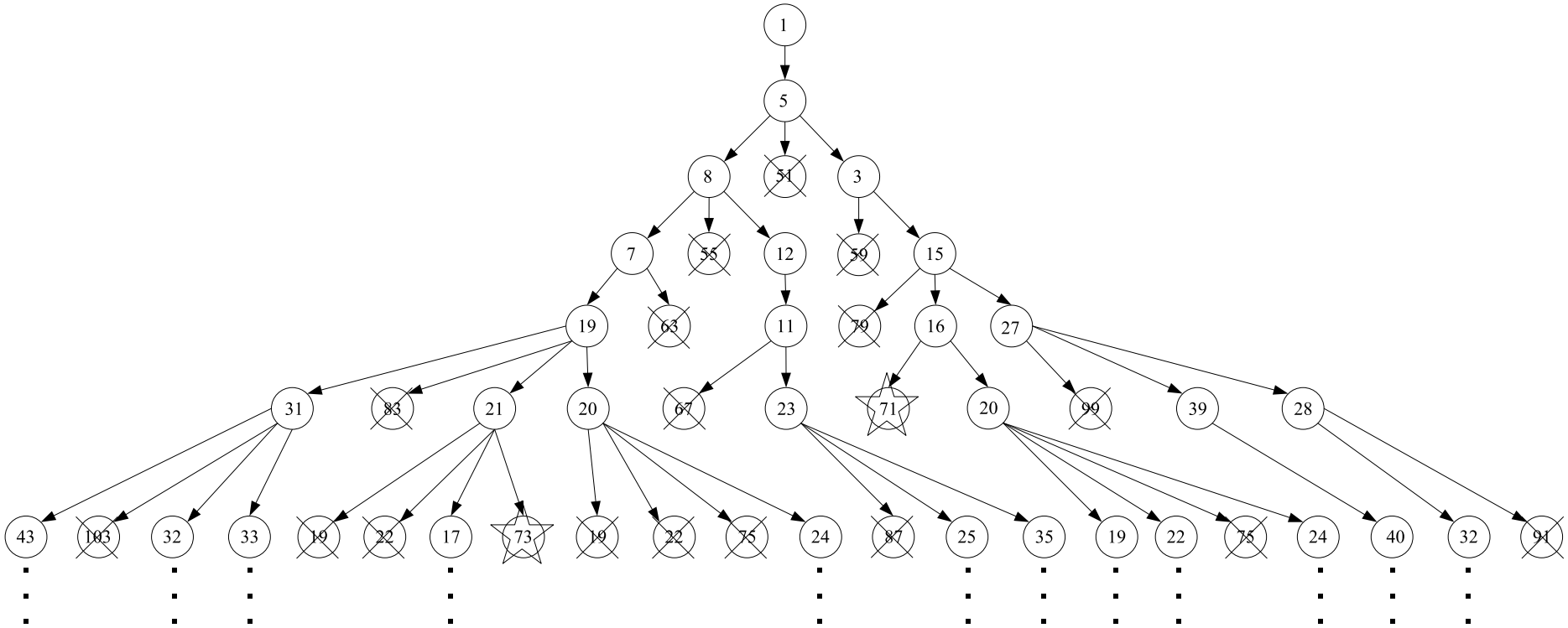


Figura 5.18: Parte da árvore de busca gerada pelo planejador no exemplo da figura 5.17.

5.3.3 Critério de planejamento utilizando a medida μ

Definidas as possíveis trajetórias para alcançar um ponto de destino, como aquelas apresentadas na tabela 5.5 para o exemplo ilustrado na figura 5.17, é preciso agora ordenar essas trajetórias, sob algum critério, possibilitando a escolha da melhor trajetória e finalizando, assim, a etapa de planejamento. Seja X_P o espaço de estados do autômato P, $\mathcal{L}(P_i)$ a linguagem gerada pelo autômato P a partir do estado i, ou seja,

$$\mathcal{L}(P_i) = \{s \in E^* : f(i,s) \in X_P\}, \tag{5.12}$$

e

$$\mathcal{L}_j = \overline{\{s_j\}},\ j = 1, 2, ..., t\ , \tag{5.13}$$

em que s_j é a *j-ésima* possível trajetória que leva o robô do estado atual i ao seu destino, $\mathcal{L}_j$ é a linguagem definida pelo fecho do prefixo da trajetória s_j e t é o número de possíveis trajetórias encontradas através do procedimento apresentado na seção 5.3.2. Assim sendo, as linguagens $\mathcal{L}_j$ respeitam a seguinte relação de inclusão:

$$\mathcal{L}_j \subset \mathcal{L}(P_i),\ j = 1, 2, ..., t\ , \tag{5.14}$$

ou seja, as linguagens $\mathcal{L}_j$ são sublinguagens de $\mathcal{L}(P_i)$. Analisando sob essa perspectiva, é possível transformar um conjunto não ordenado, ou parcialmente ordenado, de sublinguagens de $\mathcal{L}(P_i)$, em um conjunto de sublinguagens totalmente ordenado, utilizando o parâmetro de avaliação de desempenho de controladores, ou sublinguagens, μ. Para tanto, é preciso definir dois parâmetros do autômato P, apresentados no capítulo 4: o vetor característico do autômato P, $\underline{\mathcal{X}}$, e a matriz de probabilidades de ocorrência de eventos, $\widetilde{\mathbf{\Pi}}$. A partir da matriz $\widetilde{\mathbf{\Pi}}$, é construída a matriz de transição de estados, $\mathbf{\Pi}$, para cada sublinguagem $\mathcal{L}_j$ (ver capítulo 4), possibilitando assim o cálculo direto do vetor $\boldsymbol{\mu}$ através da fórmula apresentada na equação (4.60), ou seja,

$$\boldsymbol{\mu} = (\mathbf{I} - \mathbf{\Pi})^{-1}\underline{\mathcal{X}}. \tag{5.15}$$

Vetor característico (X):

A partir do desenvolvimento do autômato de planejamento P, ilustrado nas figuras 5.15(a) e 5.15(b), é possível notar que todos os estado definidos foram marcados. Essa marcação foi feita pois cada um dos estados indica a realização de uma ação de movimento pelo robô. Assim sendo, considerando a lei de formação do vetor característico $\underline{\mathcal{X}}$, deve ser associado um valor $\mathcal{X}_i$, diferente de zero, a cada estado i do autômato P, de acordo com o impacto desse estado no sistema modelado. No caso do autômato de planejamento desenvolvido, a construção do vetor característico

Tabela 5.5: Sequências de eventos que definem as possíveis trajetórias para alcançar o destino.

Rótulo da sequência	Sequência de eventos
s_1	$v_4 r_1 h_3 r_1 v_2 r_1 h_2 r_1 v_f$
s_2	$v_4 r_1 h_1 r_2 v_2 r_1 h_2 r_1 v_3 r_1 h_2 r_1 v_f$
s_3	$v_1 r_1 h_1 r_2 v_f$
s_4	$v_4 r_1 h_2 r_1 v_4 r_1 h_1 r_1 v_f$
s_6	$v_4 r_1 h_1 r_1 v_f$
s_7	$v_1 r_1 h_2 r_2 v_3 r_2 h_2 r_2 v_2 r_2 h_1 r_1 v_f$
s_8	$v_5 r_1 h_1 r_1 v_2 r_2 h_2 r_1 v_4 r_1 h_2 r_1 v_f$
s_9	$v_5 r_1 h_1 r_1 v_2 v_f$
s_{10}	$v_4 r_1 h_1 r_2 v_2 r_1 h_1 r_1 v_3 r_1 h_1 r_1 v_f$
s_{11}	$v_1 r_1 h_2 r_2 v_2 r_1 h_1 r_2 v_2 r_2 h_3 r_2 v_2 r_2 h_1 r_1 v_f$
s_{12}	$v_1 r_1 h_2 r_2 v_2 r_2 h_1 r_2 v_f$
s_{13}	$v_5 r_1 h_1 r_1 v_2 r_2 h_1 r_1 v_2 r_1 h_1 r_1 v_f$
s_{14}	$v_5 r_1 h_1 r_1 v_2 r_2 h_1 r_2 v_2 r_1 h_1 r_1 v_3 r_1 h_2 r_1 v_f$
s_{15}	$v_1 r_1 h_3 r_2 v_2 r_2 h_2 r_2 v_f$
s_{16}	$v_4 r_1 h_1 r_2 v_2 r_1 h_1 r_1 v_2 r_2 h_1 r_1 v_2 r_1 h_4 r_1 v_f$
s_{17}	$v_1 r_1 h_3 r_2 v_2 r_2 h_1 r_1 v_2 r_2 h_2 r_2 v_2 r_2 h_1 r_1 v_f$
s_{18}	$v_1 r_1 h_2 r_2 v_3 r_2 h_1 r_2 v_2 v_f$
s_{19}	$v_1 r_1 h_2 r_1 v_2 r_1 h_1 r_2 v_f$
s_{20}	$v_1 r_1 h_2 r_1 v_3 r_1 h_1 r_1 v_f$
s_{21}	$v_1 r_1 h_3 r_2 v_3 r_2 h_2 r_2 v_2 v_f$
s_{22}	$v_4 r_1 h_1 r_2 v_2 r_1 h_2 r_1 v_2 r_1 h_1 r_2 v_2 r_1 h_1 r_1 v_f$
s_{23}	$v_5 r_1 h_3 r_1 v_2 r_1 h_2 r_2 v_f$
s_{24}	$v_5 r_1 h_3 r_1 v_3 r_1 h_2 r_1 v_f$
s_{25}	$v_1 r_1 h_3 r_2 v_3 r_2 h_1 r_2 v_2 r_1 h_1 r_2 v_f$
s_{26}	$v_1 r_1 h_2 r_2 v_2 r_1 h_1 r_2 v_2 r_2 h_2 r_2 v_2 v_f$
s_{27}	$v_1 r_1 h_3 r_2 v_2 r_2 h_1 r_1 v_2 r_2 h_1 r_2 v_2 v_f$
s_{28}	$v_1 r_1 h_3 r_2 v_3 r_2 h_3 r_2 v_2 r_3 h_1 r_1 v_f$
s_{29}	$v_5 r_1 h_2 r_1 v_2 r_2 h_1 r_1 v_2 r_1 h_2 r_1 v_f$
s_{30}	$v_5 r_1 h_3 r_1 v_2 r_1 h_1 r_2 v_2 r_1 h_1 r_1 v_f$
s_{31}	$v_5 r_1 h_3 r_1 v_3 r_1 h_1 r_1 v_2 r_2 h_1 r_2 v_f$

$\underline{\mathcal{X}}$ segue a seguinte lei de formação:

$$\mathcal{X}_i = \begin{cases} 0, & \text{se } i = x_r; \\ 1, & \text{se } i = x_f; \\ \tau < 0, & \text{do contrário.} \end{cases} \tag{5.16}$$

em que x_r é o estado do robô no autômato P no momento da solicitação do planejamento, x_f é o estado do autômato P que corresponde ao destino especificado para o robô e τ é um valor negativo, para indicar os estados marcados indesejáveis. Para o sistema de planejamento proposto adotou-se $\tau = -1 \times 10^{-4}$. Esse valor foi alterado em diversos experimentos, para valores menores que zero e maiores que -1×10^{-2}, sem, no entanto, apresentar alterações na trajetória escolhida pelo planejamento, .

Matriz de probabilidades de ocorrência de eventos ($\widetilde{\mathbf{\Pi}}$):

Para definir os elementos da matriz $\widetilde{\mathbf{\Pi}}$, segundo os resultados apresentados em [14–16, 47–49], é preciso observar o funcionamento da planta do sistema e, a parir da observação do número de ocorrências de um evento em um dado estado, definir a probabilidade de ocorrência desse evento, ou seja, $\tilde{\pi}$. No caso deste trabalho, o índice μ está sendo utilizado, não para avaliar um controlador, mas para definir a melhor trajetória que o robô deve executar para realizar uma dada tarefa. Assim sendo, o autômato P não representa uma planta física, e sim o modelo de todos os possíveis movimentos que o robô pode executar no ambiente. Dessa forma, a probabilidade de ocorrência de um evento depende da posição inicial do robô e do ponto de destino do mesmo, que não podem ser determinados *a priori*. Para contornar esse problema foi considerado que as chances do robô, estando em um estado qualquer x, executar um evento qualquer e_1 é a mesma de executar um outro evento e_2, desde que $\{e_1, e_2\} \subset \Gamma(x)$, e que as chances do robô executar um evento e_3 é zero se $e_3 \notin \Gamma(x)$. Dessa forma, foi definida a seguinte lei de formação para os elementos da matriz $\widetilde{\mathbf{\Pi}}$:

$$\tilde{\pi}_{ij} = \begin{cases} \varsigma, & \text{se } e_j \in \Gamma(x_i); \\ 0, & \text{se } e_j \notin \Gamma(x_i), \end{cases} \tag{5.17}$$

para qualquer $e_j \in E$ e qualquer $x_i \in X_p$. O valor de ς utilizado nesse trabalho foi 0,2 para respeitar a formalização da medida de linguagem μ (ver capítulo 4), que diz que

$$\sum_j \tilde{\pi}_{ij} < 1, \ \forall x_i \in X_p \text{ e } \forall e_j \in E. \tag{5.18}$$

Como em todos os estados $x_i \in X_p$, o maior número de eventos pertencentes a $\Gamma(x_i)$ é quatro, o resultado do somatório apresentado na equação (5.18), para $\varsigma = 0,2$, é sempre menor que um, o que justifica o valor adotado. O valor de ς foi alterado

em diversos experimentos, sempre mantendo o limite imposto pela equação (5.18), ou seja, $\varsigma < 0,25$. Os resultados obtidos para a trajetória escolhida foram sempre os mesmos, alterando apenas o valor absoluto da medida de linguagem μ para cada uma das trajetórias candidatas.

Com os valores de $\underline{\mathcal{X}}$ e $\widetilde{\mathbf{\Pi}}$ determinados para a matriz P, é possível utilizar a equação (5.15) para calcular, de forma algébrica, o valor do vetor $\boldsymbol{\mu}$ para todas as sublinguagens de $\mathcal{L}(P_i)$. A partir do vetor $\boldsymbol{\mu}$, é possível colocar as trajetórias candidatas (a levarem o robô do estado atual ao estado de destino) em ordem, segundo o critério de desempenho apresentado. Esse ordenamento, que será o resultado do planejamento, possibilita a escolha da melhor trajetória, ou seja, a trajetória que deverá ser executada pelo robô. Para comparar as trajetórias planejadas é utilizado o elemento μ_r do vetor $\boldsymbol{\mu}$, que representa as palavras que partem do estado x_r (estado atual do robô no inicio do planejamento).

Retornando ao exemplo ilustrado na figura 5.17, cujas possíveis trajetórias encontradas estão listadas na tabela 5.5, é possível calcular os vetores $\boldsymbol{\mu}$ que representam cada uma das linguagens $\mathcal{L}_j = \overline{s_j}$. Utilizando a medida μ_1 (uma vez que o robô parte do estado 1) de cada uma das 32 linguagens avaliadas, é possível ordenar essas linguagens, conforme visto na tabela 5.6. Os valores de μ_1 apresentados na tabela 5.6 são normalizados, para melhor visualização e comparação. Dessa forma, é possível concluir que a melhor trajetória é aquela definida pela sequência cujo valor de μ_1 é igual a 1, ou seja, $s_3 = v_1 r_1 h_1 r_2 v_f$.

5.4 Resultados obtidos a partir de simulações

O sistema de navegação proposto nesse trabalho é formado pelo supervisor apresentado na seção 5.2 e pelo planejador descrito na seção 5.3. Para implementar esse sistema foi utilizado o *software* livre MobileSim (figura 5.19), disponibilizado gratuitamente pela *Mobile Robots Inc.*. Esse programa simula o ambiente de navegação e o robô P3-DX, que é a plataforma móvel considerada. O servidor utilizado para realizar a conexão entre o sistema de navegação e o robô (ou o simulador) foi desenvolvido pelo Prof. Emanuel Slawinski do *Instituto de Automática* (INAUT) da *Universidad Nacional de San juan*, na Argentina. Esse servidor, disponibilizado em código aberto, foi desenvolvido em *C++*. Seu código foi alterado neste trabalho para permitir a interface do operador com o sistema de navegação proposto, sendo a interface desenvolvida apresentada na figura 5.20.

Para ilustrar o funcionamento do sistema de navegação proposto serão utilizados dois exemplos considerando os pontos I (inicial) e F (final), indicados na figura 5.21: (*i*) O robô parte do ponto I com destino ao ponto F, sem obstáculo e; (*ii*) O robô pare do ponto I com destino ao ponto F, com obstáculo (figura 5.21).

Tabela 5.6: Sequências de eventos, ordenadas segundo μ_1, que definem as possíveis trajetórias para alcançar o destino.

Rótulo da sequência	Sequência de eventos	μ_1 (normalizado)
s_3	$v_1r_1h_1r_2v_f$	1,0000
s_6	$v_4r_1h_1r_1v_f$	0,5000
s_9	$v_5r_1h_1r_1v_2v_f$	0,1250
s_{12}	$v_1r_1h_2r_2v_2r_2h_1r_2v_f$	0,0312
s_4	$v_4r_1h_2r_1v_4r_1h_1r_1v_f$	0,0156
s_{15}	$v_1r_1h_3r_2v_2r_2h_2r_2v_f$	0,0078
s_{18}	$v_1r_1h_2r_2v_3r_2h_1r_2v_2v_f$	0,0078
s_{19}	$v_1r_1h_2r_1v_2r_1h_1r_2v_f$	0,0078
s_1	$v_4r_1h_3r_1v_2r_1h_2r_1v_f$	0,0039
s_{20}	$v_1r_1h_2r_1v_3r_1h_1r_1v_f$	0,0039
s_{21}	$v_1r_1h_3r_2v_3r_2h_2r_2v_2v_f$	0,0019
s_{23}	$v_5r_1h_3r_1v_2r_1h_2r_2v_f$	0,0019
s_{10}	$v_4r_1h_1r_2v_2r_1h_1r_1v_3r_1h_1r_1v_f$	0,0010
s_{13}	$v_5r_1h_1r_1v_2r_2h_1r_1v_2r_1h_1r_1v_f$	0,0010
s_{24}	$v_5r_1h_3r_1v_3r_1h_2r_1v_f$	0,0010
s_7	$v_1r_1h_2r_2v_3r_2h_2r_2v_2r_2h_1r_1v_f$	0,0005
s_{25}	$v_1r_1h_3r_2v_3r_2h_1r_2v_2r_1h_1r_2v_f$	0,0005
s_{26}	$v_1r_1h_2r_2v_2r_1h_1r_2v_2r_2h_2r_2v_2v_f$	0,0005
s_{27}	$v_1r_1h_3r_2v_2r_2h_1r_1v_2r_2h_1r_2v_2v_f$	0,0005
s_2	$v_4r_1h_1r_2v_2r_1h_2r_1v_3r_1h_2r_1v_f$	0,0002
s_5	$v_4r_1h_2r_2v_4r_1h_1r_1v_3r_1h_2r_1v_f$	0,0002
s_8	$v_5r_1h_1r_1v_2r_2h_2r_1v_4r_1h_2r_1v_f$	0,0002
s_{29}	$v_5r_1h_2r_1v_2r_2h_1r_1v_2r_1h_2r_1v_f$	0,0002
s_{30}	$v_5r_1h_3r_1v_2r_1h_1r_2v_2r_1h_1r_1v_f$	0,0002
s_{11}	$v_1r_1h_2r_2v_2r_1h_1r_2v_2r_2h_3r_2v_2r_2h_1r_1v_f$	0,0001
s_{16}	$v_4r_1h_1r_2v_2r_1h_1r_1v_2r_2h_1r_1v_2r_1h_4r_1v_f$	0,0001
s_{17}	$v_1r_1h_3r_2v_2r_2h_1r_1v_2r_2h_2r_2v_2r_2h_1r_1v_f$	0,0001
s_{22}	$v_4r_1h_1r_2v_2r_1h_2r_1v_2r_1h_1r_2v_2r_1h_1r_1v_f$	0,0001
s_{28}	$v_1r_1h_3r_2v_3r_2h_3r_2v_2r_3h_1r_1v_f$	0,0001
s_{31}	$v_5r_1h_3r_1v_3r_1h_1r_1v_2r_2h_1r_2v_f$	0,0001
s_{14}	$v_5r_1h_1r_1v_2r_2h_1r_2v_2r_1h_1r_1v_3r_1h_2r_1v_f$	0,0000

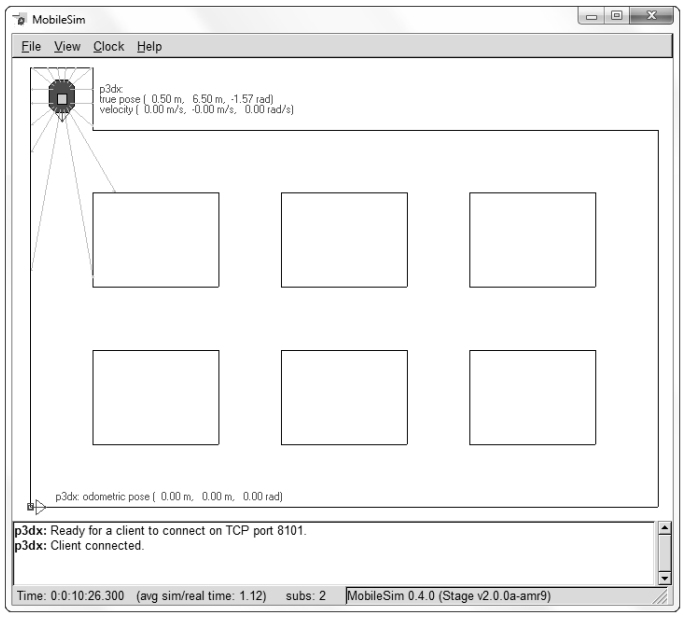


Figura 5.19: Simulador MobileSim, disponibilizado pela *Mobile Robots Inc.* gratuitamente.

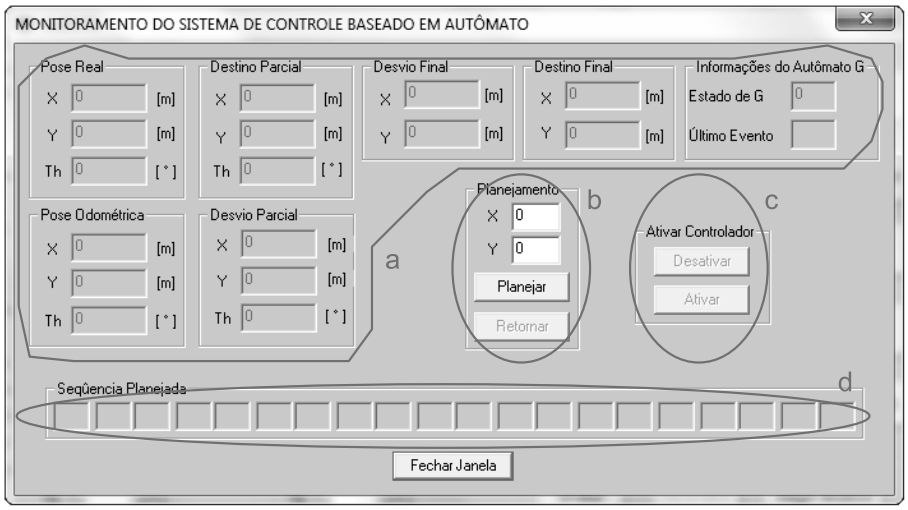


Figura 5.20: Interface de operação do sistema de navegação: (a) Variáveis do sistema; (b) Controle de planejamento; (c) Controle de ativação/desativação da navegação; (d) Última sequência gerada pelo planejador.

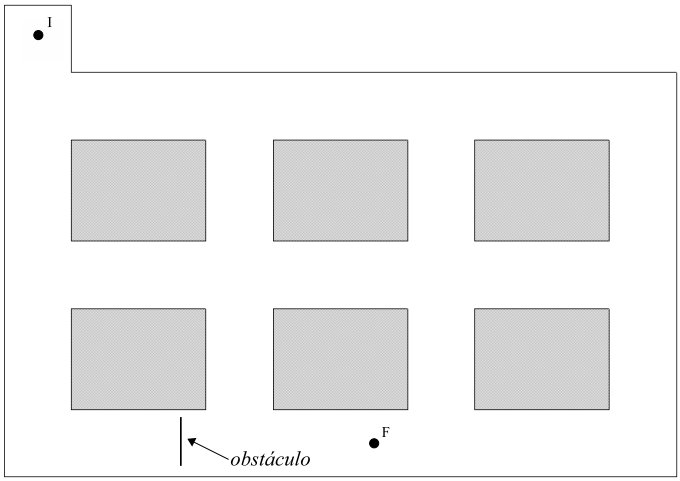


Figura 5.21: Ponto inicial I e ponto de destino F para os experimentos 1 e 2 e o obstáculo do experimento 2.

Tabela 5.7: As 5 primeiras sequências de eventos, ordenadas segundo μ_1, que definem as possíveis trajetórias para alcançar o destino F, partindo do ponto I.

Rótulo da sequência	Sequência de eventos	μ_1 (normalizado)
s_1	$v_5 r_1 h_1 h_f$	1,0000
s_2	$v_1 r_1 h_1 r_2 v_3 r_1 h_f$	0,2500
s_3	$v_4 r_1 h_1 r_2 v_2 r_1 h_f$	0,2500
s_4	$v_1 r_1 h_2 r_2 v_3 r_2 h_f$	0,1249
s_5	$v_4 r_1 h_2 r_2 v_2 r_2 h_f$	0,1249

Experimento 1: Navegação sem obstáculo

Nesse experimento, a posição $x = 5,5$ e $y = 0,5$ (ponto F) foi definida como destino para o robô, por intermédio da interface de operação e, em seguida, ativado o planejamento. Um vez ativado, o sistema de planejamento escolhe, dentre as possíveis trajetórias calculadas utilizando a metodologia descrita na seção anterior, a sequência s_1, apresentada na tabela 5.7, como a sequência a ser executada pelo robô para alcançar o destino especificado. Na tabela 5.7 são apresentadas as cinco melhores trajetórias escolhidas pelo sistema de planejamento, em ordem de prioridade, de acordo com o critério de planejamento μ. A trajetória s_1, executada pelo robô, é ilustrada na figura 5.22. Como não existem obstáculos no caminho planejado para o robô, a execução da tarefa ocorre normalmente, com o robô alcançando o destino sem a necessidade de replanejar a tarefa.

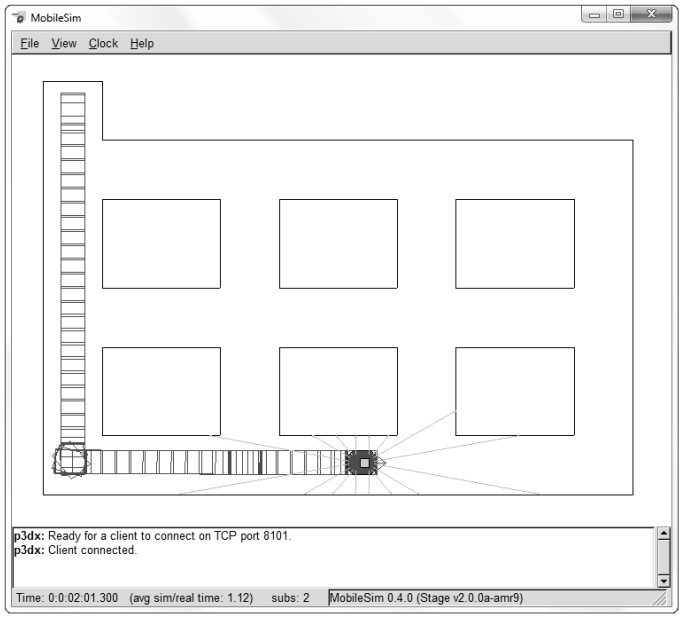


Figura 5.22: Trajetória executada pelo robô no experimento 1.

Experimento 2: Navegação com obstáculo

Nesse experimento a tarefa é a mesma do experimento anterior, sendo a única diferença o obstáculo colocado no caminho a ser executado pelo robô. Quando o operador define a posição $x = 5,5$ e $y = 0,5$ (ponto F) como destino, a resposta do sistema de planejamento é a mesma do exemplo anterior, apresentada na tabela 5.7. O robô inicializa a execução da trajetória planejada normalmente até o momento em que um obstáculo é detectado (figura 5.23), quando ocorre o evento *od*. Nesse momento, em virtude do comportamento modelado pela especificação 2 na seção 5.2.3, o robô recebe o comando para interromper a execução da tarefa (evento *pr*) e, após parar completamente (evento v_0), ele retorna ao último ponto de transição válido (palavra $r_3w_0h_rv_0$). Após a ocorrência desses eventos o robô chega à posição R (estado 14 do autômato de planejamento P), indicada na figura 5.24. Nessa posição o sistema de planejamento é acionado novamente para replanejar a trajetória, definindo, desta vez, a sequência de eventos que levará o robô da posição atual R até o ponto de destino especificado F. Para evitar que, no replanejamento, o robô seja mandado novamente na direção do obstáculo, o evento r_3 (rotação de 180 graus) é desabilitado, garantindo que, no caminho replanejado, o robô não passará pelo corredor onde se encontra o obstáculo. No entanto, nenhuma informação é guardada na memória para ser utilizada em navegações futuras, ou seja, um obstáculo detectado irá afetar apenas o próximo planejamento de trajetória (replanejamento para evitar o caminho bloqueado), sendo sua posição descartada após esse planejamento.

Na tabela 5.8 são apresentadas as cinco melhores trajetórias escolhidas pelo

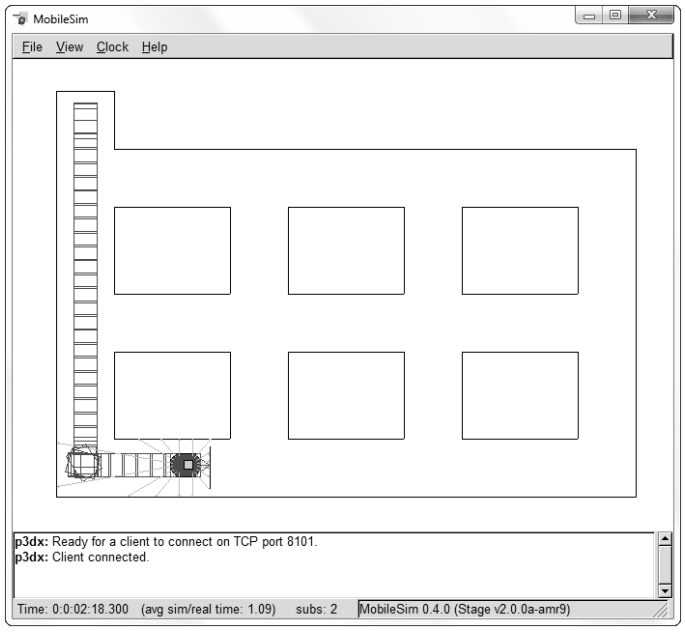


Figura 5.23: Trajetória executada pelo robô no experimento 2, até a detecção de um obstáculo.

sistema de planejamento, ordenadas segundo o critério de planejamento μ_{14} (o índice 14 é determinado pelo estado do robô no autômato de planejamento P). A trajetória escolhida para ser executada pelo robô é aquela definida pela sequência s_1 da tabela 5.8, cujo resultado obtido é ilustrada na figura 5.25. Como na nova trajetória não existem obstáculos no caminho do robô, a execução da tarefa ocorre normalmente, com o robô alcançando o destino especificado.

Depois que o robô cumpre o seu objetivo de chegar ao ponto de destino (evento tr) a tarefa é dada por encerrada. No entanto, o robô ainda precisa voltar ao ponto de partida para entregar o item que foi retirado do depósito. A tarefa de retorno, no sistema de navegação proposto, nada mais é do que uma nova tarefa de navegação independente da primeira, na qual o ponto de partida é o estado atual do robô e o ponto de destino é o estado 1 do autômato de planejamento P. Dessa forma, todo procedimento de planejamento será reinicializado para as novas coordenadas, sendo descartadas todas as informações obtidas na navegação anterior como, por exemplo, a posição do obstáculo. Isso é feito porque um obstáculo que foi encontrado na tarefa de navegação anterior não, necessariamente, estará lá na próxima tarefa de navegação.

Diversos experimentos, com diferentes pontos de destino e diferentes obstáculos, foram realizados para validar o sistema de navegação proposto nesse trabalho. A exemplo dos resultados obtidos nos experimentos 1 e 2, os resultados obtidos em todos os experimentos de validação foram satisfatórios e demonstraram o correto funcionamento do sistema proposto em todas as suas etapas.

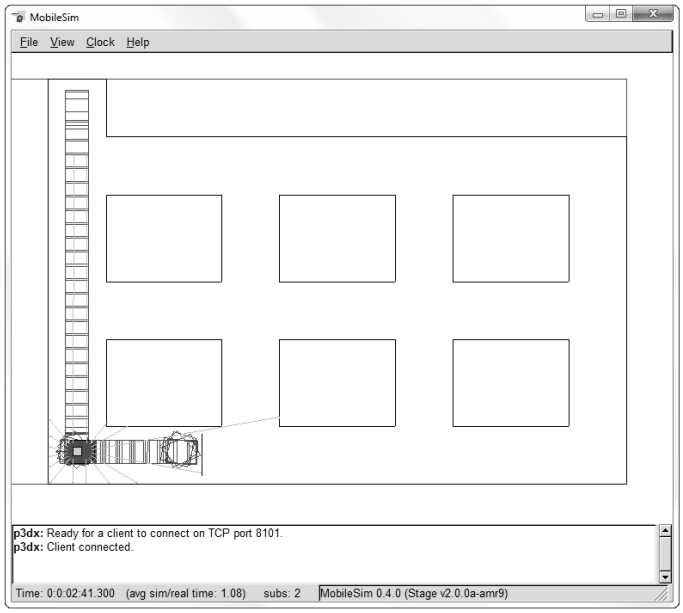


Figura 5.24: Retorno ao último ponto de transição válido após detectar um obstáculo no experimento 2.

Tabela 5.8: As 5 primeiras sequências de eventos, ordenadas segundo μ_{14}, que definem as possíveis trajetórias para alcançar o destino F, partindo do ponto R.

Rótulo da sequência	Sequência de eventos	μ_{14} (normalizado)
s_1	$r_2v_2r_2h_1r_2v_2r_1h_f$	1,0000
s_2	$r_2v_2r_2h_2r_2v_2r_2h_f$	0,4999
s_3	$r_2v_3r_2h_1r_2v_3r_1h_f$	0,2498
s_4	$r_2v_2r_2h_3r_2v_2r_2h_1h_f$	0,1248
s_5	$r_2v_3r_2h_2r_2v_3r_2h_f$	0,1248

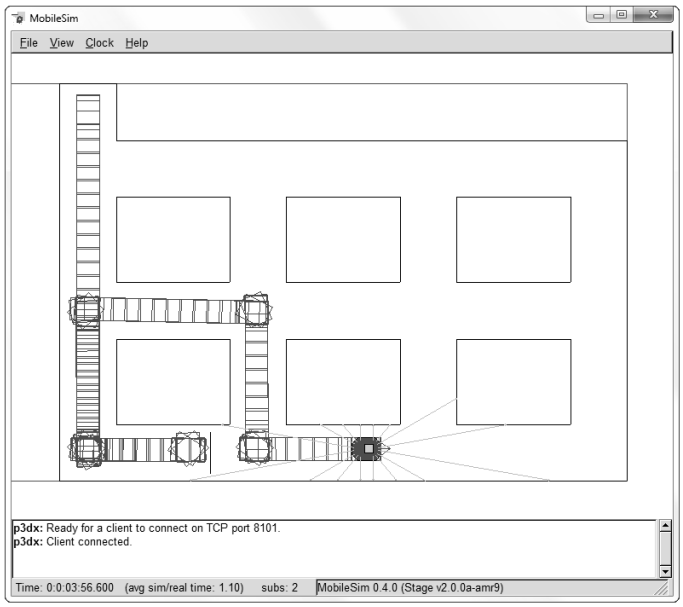


Figura 5.25: Trajetória completa executada pelo robô no experimento 2.

Capítulo 6

Conclusões e trabalhos futuros

Nesse trabalho foi apresentado o desenvolvimento e a implementação de uma arquitetura de navegação predominantemente deliberativa utilizando coordenação de comportamentos modelados por autômatos e um critério de planejamento baseado na medida de linguagem μ.

A utilização de coordenação de comportamentos na construção de uma arquitetura de navegação é particularmente interessante pois possibilita a elaboração de estratégias de navegação com complexidade elevada através da composição de vários comportamentos mais simples, como os que foram modelados nesse trabalho.

O formalismo adotado na modelagem dos comportamentos desejados para o robô foi a representação de modelos de sistemas a eventos discretos através de autômatos. A teoria de autômatos foi escolhida por fornecer ferramentas bem definidas de composição (no caso desse trabalho a composição paralela) capazes de capturar a interação entre os comportamentos desenvolvidos de forma simples e rápida, permitindo ainda a inclusão posterior de novos comportamentos no sistema, sem que para isso fosse necessária qualquer alteração nos demais comportamentos já modelados. O sistema controlado final funcionou conforme esperado, garantindo a correta execução da sequência de eventos planejada e solicitando o replanejamento da trajetória sempre que um obstáculo tenha sido detectado no caminho do robô.

O sistema de planejamento de trajetória, proposto nesse trabalho, utilizou um modelo em sistemas a eventos discretos (autômato de planejamento P) do ambiente de navegação e das ações do robô nesse ambiente para representar todas os possíveis movimentos do robô. A linguagem gerada pelo autômato de planejamento, $\mathcal{L}(P)$, foi utilizada na obtenção do vetor característico ($\underline{\mathcal{X}}$) e da matriz de probabilidade de ocorrência de eventos ($\widetilde{\mathbf{\Pi}}$), necessários para a construção da medida de linguagem μ. Considerando as trajetórias que levam o robô de um ponto inicial I a um ponto de destino F como sendo sublinguagens de $\mathcal{L}(P)$, foi possível utilizar o parâmetro μ como medida de desempenho dessas trajetórias e, dessa forma, escolher a melhor trajetória segundo o critério adotado. Esse método de planejamento dis-

creto é a principal contribuição desse trabalho por ser robusto, no sentido de que sempre irá apresentar uma trajetória (se existir alguma) para alcançar o ponto de destino, e ótimo, pois a trajetória planejada é sempre aquela que otimiza o critério de planejamento μ. Esse sistema foi testado exaustivamente para diferentes pontos de partida e de destino, apresentando sempre uma solução para o problema de navegação considerado.

Assim como qualquer arquitetura de navegação deliberativa, o conhecimento prévio do ambiente é um fator limitante do sistema proposto, sendo impossível a sua implementação sem essa informação *a priori*. Uma outra dificuldade na implementação desse trabalho é que, ao discretizar os movimentos do robô, os modelos desenvolvidos ficam extremamente conectados com os movimentos permitidos para o robô e com a estrutura do ambiente de navegação. Isso quer dizer que, para que o robô possa executar outros tipos de movimentos, é preciso reconstruir o modelo do sistema de controle final. O mesmo aconteceria para o caso de se utilizar esse sistema em um outro ambiente de navegação, o que acarretaria a reconstrução do modelo do autômato de planejamento P.

Como trabalho futuro propõe-se a generalização da técnica de planejamento desenvolvida, possibilitando a utilização do método de planejamento proposto em ambientes quaisquer, sem a necessidade de se construir manualmente um autômato de planejamento P para cada ambiente de navegação considerado. Uma outra modificação interessante é a inserção de memória no sistema de navegação, realizando a atualização do mapa do ambiente sempre que um obstáculo for detectado. A realização de uma análise da complexidade computacional do algoritmo 5.1, utilizado para encontrar as trajetórias candidatas no sistema de planejamento, é também um aspecto interessante a ser considerado futuramente. Um outro possível trabalho futuro é a implementação do sistema proposto em ambientes reais com robôs físicos e, através da realização de experimentos, incluir no modelo do sistema, eventos de falhas de sensoriamento e atuação para tornar a resposta do sistema modelado mais robusta e previsível.

Referências Bibliográficas

[1] PIRJANIAN, P. *An Overview of System Architecture for Action Selection in Mobile Robotics*. Relatório técnico, Laboratory of Image Analysis: Aalborg University, 1997.

[2] ARKIN, R. *Behavior-Based Robotics (Intelligent Robotics and Autonomous Agents)*, v. I. MIT Press, 1998.

[3] PIRJANIAN, P. *Behavior Coordination Mechanisms - State-of-the-art*. Relatório técnico, Institute of Robotics and Intelligent Systems, School of Engineering, University of Southern California, October 1999.

[4] CASSANDRAS, C. G., LAFORTUNE, S. *Introduction to Discrete Event Systems*. Springer-Verlag New York, Inc., 2006. ISBN: 0387333320.

[5] KOBAYASHI, K., USHIO, T. "An application of LLP supervisory control with Petri net models inmobile robots". In: *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*, v. 4, pp. 3015–3020, 2000.

[6] KOSECKÁ, J., BAJCSY, R. "Discrete Event Systems for autonomous mobile agents", *Robotics and Autonomous Systems*, v. 12, pp. 187–198, 1993.

[7] KOSECKÁ, J., CHRISTENSEN, H. I., BAJCSY, R. "Discrete event modeling of visually guided behaviors", *International Journal of Computer Vision*, v. 14, pp. 179–191, 1995.

[8] KOSECKÁ, J., BOGONI, L. "Application of Discrete Events Systems for Modeling and Controlling Robotic Agents". In: *Proceedings of IEEE International Conference on Robotics and Automation*, pp. 2257–2262, 1994.

[9] KOSECKÁ, J., ABDALLAH, H. B. "An Automaton Based Algebra for Specifying Robotic Agents". In: *Proceedings of the AMAST Workshop on Real-Time Systems*, 1996.

[10] KOSECKÁ, J., CHRISTENSENZ, H. I., BAJCSY, R. "Experiments in Behavior Composition". In: *Proceedings of the International Symposium on Intelligent Robotic Systems*, pp. 287–298, 1997.

[11] KOSECKÁ, J. *Supervisory Control Theory For Autonomous Mobile Agents*. Tese de Doutorado, Computer Science School from University of Pennsylvania, 1996.

[12] RAMADGE, P. J. G., WONHAM, W. M. "Supervisory control of a class of discrete event process", *SIAM Journal of Control Optim.*, v. 25, n. 1, pp. 206 – 230, 1987.

[13] RAMADGE, P. J. G., WONHAM, W. M. "The control of discrete event systems", *Proceedings of the IEEE*, v. 77, n. 1, pp. 81 – 98, 1989.

[14] RAY, A., PHOHA, S. "A language measure for discrete-event automata". In: IFAC (Ed.), *Proceedings of 15th Triennial World Congress*, 2002.

[15] RAY, A., PHOHA, S. "Signed real measure of regular languages for discrete-event automata", *International Journal of Control*, v. 76, pp. 1800 – 1808, 2003.

[16] RAY, A., PHOHA, V. V., PHOHA, S. *Quantitative Measure for Discrete Event Supervisory Control*. Springer Science + Business Media, 2005.

[17] WANG, X., FU, J., LEE, P., et al. "Robot Behavioral Selection Using Discrete Event Language Measure". In: *Proceedings of the 2004 American Control Conference*, 2004.

[18] WANG, X., LEE, P., RAY, A., et al. "Discrete Event Supervisory Control of a Mobile Robotic System", *Quantitative Measure for Discrete Event Supervisory Control*, pp. 133 – 156, 2005.

[19] MALLAPRAGADA, G., CHATTOPADHYAY, I., RAY, A. "Autonomous Navigation of Mobile Robots Using Optimal Control of Finite State Automata". In: IEEE (Ed.), *Proceedings of the 45th IEEE Conference on Decision and Control*, 2006.

[20] WANG, X., RAY, A., LEE, P., et al. "Optimal control of robot behavior using language measure", *Quantitative Measure for Discrete Event Supervisory Control*, pp. 157 – 182, 2005.

[21] TYRRELL, T. *Computational Mechanisms for Action Selection*. Tese de Doutorado, University of Edinburgh, 1993.

[22] MAES, P. "How to do the right thing", *Connection Science Journal*, v. 1, pp. 291–323, 1989.

[23] BAILEY, T., DURRANT-WHYTE, H. "Simultaneous localisation and mapping (SLAM): Part I - The essential algorithms", *IEEE Robotics and Automation Magazine*, v. 13, n. 2, 2006.

[24] BAILEY, T., DURRANT-WHYTE, H. "Simultaneous localisation and mapping (SLAM): Part II - State of the art", *IEEE Robotics and Automation Magazine*, v. 13, n. 3, 2006.

[25] LAIRD, J. E., NEWELL, A., ROSENBLOOM, P. S. "SOAR: an architecture for general intelligence", *Artif. Intell.*, v. 33, n. 1, pp. 1–64, 1987. ISSN: 0004-3702.

[26] BICHO, E., MALLET, P., SCHONER, G. "Using attractor dynamics to control autonomous vehicle motion", *Industrial Electronics Society, 1998. IECON '98. Proceedings of the 24th Annual Conference of the IEEE*, v. 2, pp. 1176–1181 vol.2, 1998. doi: 10.1109/IECON.1998.724266.

[27] ANDERSEN, C. S., CHRISTENSEN, H. I., KIRKEBY, N. O. S., et al. "A System for VIsion Supported NAVigation". In: *Proceedings of Nordic Summer School on Active Vision and Geometric Modeling*, pp. 251–258, 1992.

[28] CURRIE, K., TATE, A., BRIDGE, S. "O-plan: the open planning architecture", *Artificial Intelligence*, v. 52, pp. 49–86, 1991.

[29] KOSAKA, A., KAK, A. "Fast Vision-guided Mobile Robot Navigation Using Model-based Reasoning And Prediction Of Uncertainties", *Intelligent Robots and Systems, 1992., Proceedings of the 1992 lEEE/RSJ International Conference on*, v. 3, pp. 2177–2186, Jul 1992. ISSN: 1.

[30] MORAVEC, H. "The Stanford Cart and the CMU Rover", *Proceedings of the IEEE*, v. 71, n. 7, pp. 872–884, July 1983. ISSN: 0018-9219.

[31] BROOKS, R. "A robust layered control system for a mobile robot", *Robotics and Automation, IEEE Journal of*, v. 2, n. 1, pp. 14–23, Mar 1986. ISSN: 0882-4967.

[32] FREIRE, E., BASTOS-FILHO, T., SARCINELLI-FILHO, M., et al. "A control architecture for mobile robots using fusion of the output of distinct controllers", *Intelligent Control, 2002. Proceedings of the 2002 IEEE International Symposium on*, pp. 142–147, 2002. doi: 10.1109/ISIC.2002.1157753.

[33] FREIRE, E. *Controle de Robôs Móveis por Fusão de Sinais de Controle Usando Filtro de Informação Descentralizado.* Tese de Doutorado, Universidade Federal do Espírito Santo, Vitória-ES, Agosto 2002.

[34] ARKIN, R. "Motor schema based navigation for a mobile robot: An approach to programming by behavior", *Robotics and Automation. Proceedings. 1987 IEEE International Conference on*, v. 4, pp. 264–271, Mar 1987.

[35] FREIRE, E., BASTOS-FILHO, T., SARCINELLI-FILHO, M., et al. "A new mobile robot control approach via fusion of control signals", *Systems, Man, and Cybernetics, Part B, IEEE Transactions on*, v. 34, n. 1, pp. 419–429, Feb. 2004. ISSN: 1083-4419. doi: 10.1109/TSMCB.2003.817034.

[36] ROSENBLATT, J. "DAMN: A Distributed Architecture for Mobile Navigation". In: *Journal of Experimental and Theoretical Artificial Intelligence*, pp. 339–360. AAAI Press, 1997.

[37] CONNELL, J. H. "SSS: A hybrid architecture applied to robot navigation". In: *In Proceedings of the 1992 IEEE International Conference on Robotics and Automation*, pp. 2719–2724, 1992.

[38] GAT, E. "Integrating planning and reaction in a heterogeneous asynchronous architecture for mobile robot navigation", *SIGART Bull.*, v. 2, n. 4, pp. 70–74, 1991. ISSN: 0163-5719. doi: 10.1145/122344.122357.

[39] SAFFIOTTI, A., KONOLIGE, K., RUSPINI, E. H. "A Multivalued Logic Approach to Integrating Planning and Control", *Artificial Intelligence*, v. 76, pp. 481–526, 1995.

[40] ARKIN, R. C. "Towards the unification of navigational planning and reactive control". In: *In AAAI Spring Symposium on Robot Navigation*, pp. 1–5, 1989.

[41] ARKIN, R. C., BALCH, T. "AuRA: Principles and Practice in Review", *Journal of Experimental and Theoretical Artificial Intelligence*, v. 9, pp. 175–189, 1997.

[42] BLAASVAER, H., PIRJANIAN, P., CHRISTENSEN, H. I. "An autonomous mobile robot navigation system". In: *in Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics*, pp. 2266–2271, 1994.

[43] FERREIRA, A., PEREIRE, F. G., VASSALO, R. F., et al. "An approach to avoid obstacles in mobile robot navigation: the tangential escape", *SBA: Controle e Automação*, v. 19, pp. 395 – 405, 2008.

[44] PIRJANIAN, P. "Multiple objective behavior-based control", *Robotics and Autonomous Systems*, v. 31, n. 1-2, pp. 53 – 60, 2000. ISSN: 0921-8890. doi: 10.1016/S0921-8890(99)00081-0.

[45] CARELLI, R., SORIA, C., NASISI, O., et al. "Stable AGV corridor navigation with fused vision-based control signals", *IECON 02 [Industrial Electronics Society, IEEE 2002 28th Annual Conference of the]*, v. 3, pp. 2433–2438 vol.3, Nov. 2002.

[46] KUMAR, R., GARG, V. K. *Modeling and Control of Logical Discrete Event Systems*. MA: Kluwer Academic Publishers, 1995.

[47] WANG, X., RAY, A. "A language measure for performance evaluation of discrete-event supervisory control systems", *Applied Mathematical Modeling*, v. 28, pp. 817 – 833, 2004.

[48] WANG, X., RAY, A. "Signed real measure of regular languages". In: *Proceedings of the American Control Conference*, 2002.

[49] SURANA, A., RAY, A. "Signed real measure of regular languages". In: IEEE (Ed.), *Proceedings of IEEE Conference on Decision and Control*, 2003.