UNIVERSIDADE ESTADUAL DE CAMPINAS
Faculdade de Engenharia Elétrica e de Computação

Bruno Guedes da Silva

Sistema de memória episódica para agentes inteligentes utilizando CST e representação de conhecimento *Idea*

Campinas

2024

UNIVERSIDADE ESTADUAL DE CAMPINAS
Faculdade de Engenharia Elétrica e de Computação

Bruno Guedes da Silva

Sistema de memória episódica para agentes inteligentes utilizando CST e representação de conhecimento *Idea*

Dissertação apresentada à Faculdade de Engenharia Elétrica e de Computação da Universidade Estadual de Campinas como parte dos requisitos exigidos para a obtenção do título de Mestre em Engenharia Elétrica, na Área de Engenharia da Computação.

Orientador: Prof. Dr. Ricardo Ribeiro Gudwin

Este exemplar corresponde à versão final da tese defendida pelo aluno Bruno Guedes da Silva, e orientada pelo Prof. Dr. Ricardo Ribeiro Gudwin

_____

Campinas
2024

Ficha catalográfica
Universidade Estadual de Campinas (UNICAMP)
Biblioteca da Área de Engenharia e Arquitetura
Rose Meire da Silva - CRB 8/5974

Si38s

Silva, Bruno Guedes da, 1997-
Sistema de memória episódica para agentes inteligentes utilizando CST e representação de conhecimento *Idea* / Bruno Guedes da Silva. – Campinas, SP : [s.n.], 2024.

Orientador: Ricardo Ribeiro Gudwin.
Dissertação (mestrado) – Universidade Estadual de Campinas (UNICAMP), Faculdade de Engenharia Elétrica e de Computação.

1. Memória episódica. 2. Arquitetura cognitiva. 3. Agentes inteligentes (Software). I. Gudwin, Ricardo Ribeiro, 1967-. II. Universidade Estadual de Campinas (UNICAMP). Faculdade de Engenharia Elétrica e de Computação. III. Título.

Informações Complementares

Título em outro idioma: Episodic memory system for intelligent agents utilizing CST and *Idea* knowledge representation
Palavras-chave em inglês:
Episodic memory
Cognitive architectures
Intelligent agents
Área de concentração: Engenharia de Computação
Titulação: Mestre em Engenharia Elétrica
Banca examinadora:
Ricardo Ribeiro Gudwin [Orientador]
Paula Dornhofer Paro Costa
Leonardo Lana de Carvalho
Data de defesa: 10-07-2024
Programa de Pós-Graduação: Engenharia Elétrica

Identificação e informações acadêmicas do(a) aluno(a)
- ORCID do autor: https://orcid.org/0009-0000-5244-8403
- Currículo Lattes do autor: http://lattes.cnpq.br/4260952775077993

COMISSÃO JULGADORA - DISSERTAÇÃO DE MESTRADO

Candidato: Bruno Guedes da Silva **RA:** 203657

Data da Defesa: 10 de julho de 2024

Título da Tese: "Sistema de memória episódica para agentes inteligentes utilizando CST e representação de conhecimento *Idea*"

Prof. Dr. Ricardo Ribeiro Gudwin (Presidente, FEEC/UNICAMP)

Profa. Dra. Paula Dornhofer Paro Costa (FEEC/UNICAMP)

Prof. Dr. Leonardo Lana de Carvalho (UFVJM)

A ata de defesa, com as respectivas assinaturas dos membros da Comissão Julgadora, encontra-se no SIGA (Sistema de Fluxo de Dissertação/Tese) e na Secretaria de Pós-Graduação da Faculdade de Engenharia Elétrica e de Computação.

Agradecimentos

Esse trabalho foi realizado como parte do Hub de Inteligência Artificial e Arquiteturas Cognitivas (H.IAAC) com suporte do PPI-Softex/MCTI sob financiamento 01245.013778/2020-21 através do Governo Federal Brasileiro.

Resumo

A memória episódica é uma função cognitiva que permite que os seres humanos se lembrem de experiências passadas para resolver tarefas atuais. Devido à sua função em muitos processos cognitivos que apoiam o comportamento humano complexo, há um interesse cada vez maior no desenvolvimento de modelos computacionais de memória episódica. Em particular, foram feitas diferentes propostas considerando seu uso em arquiteturas cognitivas, com foco em diferentes aspectos da memória episódica. Nesta dissertação é proposta um modelo de arquitetura cognitiva com um sistema integrado de memória episódica para agentes inteligentes. Com base nas mudanças percebidas no ambiente e nos processos cognitivos internos do agente, o sistema realiza a segmentação temporal da experiência do agente em dois níveis: eventos e episódios. Eventos capturam a mudança temporal de propriedades dos objetos observados, enquanto episódios agregam múltiplos eventos sob um contexto interno estável do agente. Episódios são incorporados em uma memória episódica de longo prazo, na qual os elementos de eventos passados podem ser reutilizados para representar novos eventos. Por fim, um módulo de recuperação permite que os episódios sejam buscados na memória. Uma implementação piloto é desenvolvida para validação do modelo em ambientes simulados. Os resultados dos experimentos demonstram a operação correta dos módulos desenvolvidos para codificação, armazenamento e recuperação dos episódios observados.

Palavras-chaves: Memória episódica; Arquiteturas cognitivas; Agentes inteligentes

Abstract

Episodic memory is a cognitive function that allows humans to remember past experiences in order to solve current tasks. Due to its role in many cognitive processes that support complex human behavior, there is growing interest in the development of computational models of episodic memory. In particular, different proposals have been made considering its use in cognitive architectures, focusing on different aspects of episodic memory. This dissertation proposes a cognitive architecture model with an integrated episodic memory system for intelligent agents. Based on perceived changes in the environment and the agent's internal cognitive processes, the system performs temporal segmentation of the agent's experience into two levels: events and episodes. Events capture the temporal change in properties of observed objects, while episodes aggregate multiple events under a stable internal context of the agent. Episodes are incorporated into a long-term episodic memory, in which the elements of past events can be reused to represent new events. Finally, a retrieval module allows searching for episodes in memory. A pilot implementation is developed to validate the model in simulated environments. The results of the experiments demonstrate the correct operation of the modules developed for encoding, storing and retrieving the observed episodes.

Keywords: Episodic memory; Cognitive architectures; Intelligent agents

Lista de ilustrações

Sumário

1 Introdução

Cada vez mais, esforços são destinados à criação de sistemas computacionais que possam atuar em tarefas que necessitam a interação com humanos, como assistentes pessoais, tutores virtuais e atendimento ao cliente. Esses sistemas precisam interagir com situações dinâmicas, onde novas informações são apresentadas e diferentes tarefas precisam ser cumpridas. Entretanto, ainda é necessário instruí-los diretamente com o comportamento esperado dos mesmos, e por outro lado humanos são capazes de utilizar diversos processos cognitivos para coletar informações do ambiente e utilizá-las para adaptar-se a novos cenários e inferir o desenvolvimento futuro dos mesmos.

Neste cenário, arquiteturas cognitivas buscam modelar sistemas inteligentes que se comportem de forma próxima a de humanos. Tal esforço é realizado através da exploração de diferentes mecanismos para levantar evidências de quais ajudam a produzir um comportamento inteligente em agentes.

1.1 Motivação

Dentre diversas capacidades da mente humana, a memória episódica possibilita relembrarmos experiências passadas e extrair informações de o que, quando e onde ocorreu um evento, sendo o meio prevalente de memória adquirida no cotidiano humano (TSIEN, 2008). Desde a concepção do termo 'Memória Episódica' pelo psicólogo Tulving (1972), estudos nas áreas de psicologia e neurociência de suas manifestações fenomenológicas e mecanismos mostram a participação dessa capacidade em diversos processos cognitivos humanos como planejamento futuro, navegação, interação social e imaginação (CHENG *et al.*, 2016; GILBOA; MARLATTE, 2017; MAHR; CSIBRA, 2018).

Dada a relevância de seu papel no comportamento humano, mecanismos de memória episódica têm sido desenvolvidos em diferentes propostas de arquiteturas cognitivas, para obter agentes inteligentes com comportamentos mais próximos ao humano e melhor interação entre agente e humano (MéNAGER *et al.*, 2022; MARTIN *et al.*, 2022; PARK *et al.*, 2018; NUXOLL; LAIRD, 2012).

Um estudo realizado por Kotseruba e Tsotsos (2020) analisou 84 arquiteturas

cognitivas diferentes filtradas de uma lista inicial de 195 arquiteturas. Diversos problemas que requerem estudos mais aprofundados foram apontados pelos autores, dentre eles, a exploração superficial de componentes da memória episódica nas arquiteturas. Ao mesmo tempo, o estudo do mecanismo neurobiológico e psicológico da memória episódica ainda é um campo de pesquisa com alta atividade e novas descobertas são constantemente adicionadas ao conjunto de conhecimento da área (MAHR; CSIBRA, 2018; ZACKS, 2020). Dessa forma, a incorporação dos diversos processos e características da memória episódica em um sistema computacional continua sendo alvo de pesquisas. Neste trabalho, busca-se utilizar algumas das descobertas e teorias recentes sobre memória episódica na área de neurobiologia e psicologia como inspiração no desenvolvimento de um sistema de memória episódica para arquiteturas cognitivas de forma a preencher algumas das lacunas observadas em outras propostas de arquiteturas cognitivas. Em especial, o presente trabalho explora as questões: como utilizar uma representação espacial e temporal de observações do ambiente para realizar a segmentação de eventos e episódios? Como os segmentos de eventos e episódios gerados podem ser representados em alto nível para seu armazenamento e recuperação?

1.2 Objetivos

O *Cognitive Systems Toolkit* (CST) (PARAENSE *et al.*, 2016) é uma ferramenta para o desenvolvimento de arquiteturas cognitivas desenvolvida no DCA-FEEC-UNICAMP, que permite a implementação flexível de arquiteturas multi-componentes e possui inspiração em diferentes modelos cognitivos computacionais existentes. O CST propõe componentes básicos para construção de uma arquitetura, porém não impõe restrições a uma estrutura fixa dos componentes e dos processos computacionais executados nos mesmos, diferentemente da maioria das propostas de arquiteturas cognitivas da literatura.

Recentemente, um novo mecanismo padronizado para representação do conhecimento foi introduzido no CST. Esse mecanismo se baseia no conceito de *ideias computacionais*, consolidado por meio da definição da classe *Idea* no CST. Uma *Idea* representa um bloco de conhecimento abstrato e genérico que permite a representação de uma porção, simples ou complexa, de conhecimento, sobre a qual todo tipo de conhecimento poderia, em princípio, ser representado (CAMARGO *et al.*, 2022). Mantendo a proposta do CST,

essa forma de representação é flexível e não se limita a uma estrutura específica de dados, propondo somente uma organização padronizada referente à informação carregada nesses dados e sua função no conhecimento do agente.

Este trabalho tem como objetivo principal o desenvolvimento de um sistema de memória episódica para agentes inteligentes, inspirado em teorias atuais de operação da memória humana. Particulamente, a arquitetura cognitiva utilizará o CST e a nova representação de conhecimento na forma de *Ideas*, capacitando o agente com mecanismos de codificação, armazenamento e recuperação de memórias episódicas.

Os seguintes objetivos específicos são propostos para atingir o objetivo principal:

- Desenvolvimento do módulo de codificação de episódios experienciados pelo agente, através da segmentação do fluxo contínuo de dados sensoriais recebidos pelo agente em unidades discretas significativas;
- Desenvolvimento do módulo de armazenamento de memórias episódica, no qual informações de novos episódios são assimilados e acomodados em uma memória única;
- Desenvolvimento do módulo de recuperação de episódios para reconstrução de episódios passados a partir de informações parciais.

1.3 Contribuições

As principais contribuições deste trabalho são:

- O desenvolvimento de um modelo de sistema de memória episódica. O sistema segmenta eventos e episódios por dois mecanismos paralelos e utiliza representações em alto nível das transformações ocorridas para armazenamento dos mesmos, onde são incorporadas informações perceptuais e de processamento interno da mente do agente;
- A implementação de uma arquitetura cognitiva utilizando o modelo de sistema episódico citado anteriormente. O CST e representação *Ideas* são utilizados para criação dos processos e estruturas de dados necessárias para viabilizar o modelo proposto;

- Uma biblioteca para configuração e controle de simulações no software CoppeliaSim. A biblioteca atua como uma interface de comunicação entre os componentes do CST e do software CoppeliaSim, permitindo o controle de agentes virtuais no ambiente simulado.

1.4 Organização do Trabalho

Este texto é dividido em 7 capítulos. No presente capítulo são apresentadas as motivações e objetivos deste trabalho. No Capítulo 2, são introduzidos conceitos relevantes sobre memória humana e arquiteturas cognitivas utilizados no trabalho. O Capítulo 3 introduz o CST e a representação *Ideas*, apresentando seus principais componentes e forma de organização. No Capítulo 4, será descrito o modelo de sistema episódico proposto. No Capítulo 5, é apresentada a implementação da arquitetura cognitiva utilizando o sistema episódico proposto. No Capítulo 6, são descritos os cenários experimentais e os resultados obtidos. Por fim, o Capítulo 7 apresenta as principais considerações sobre o trabalho e propostas de trabalhos futuro.

2 Fundamentação Teórica

Esse trabalho propõe um modelo e implementação **computacional** de um sistema de memória episódica, porém se posiciona na intersecção de algumas áreas do conhecimento, buscando inspiração em teorias e evidências da psicologia, neurociência e filosofia para criação de estruturas de representação e processamento de informações. Diante disso, este capítulo busca introduzir ao leitor as principais definições que foram escolhidas de diferentes áreas para o desenvolvimento e implementação do sistema de memória episódica.

Inicialmente, é apresentado a definição de memória episódica humana e como a mesma se diferencia das demais memórias humanas. São apresentadas, então, algumas de suas capacidades e os processos principais que a compõe, destacando aqueles que são inspiração para os modelos e implementações computacionais desenvolvidas.

Após isso, introduzimos a área de arquiteturas cognitivas, apresentando seu grande objetivo. Destacamos, por fim, alguns dos trabalhos já propostos de arquiteturas cognitivas que exploram a memória episódica e quais as lacunas apresentadas nos mesmos.

2.1 Memória Episódica

O estudo da memória episódica como um componente cognitivo distinto e com características próprias que o diferenciam das demais formas de memória presentes na mente humana foi introduzido por Tulving (1972). A memória episódica é classificada, segundo a comumente utilizada taxonomia de Squire e Zola-Morgan (1988) apresentada na Figura 2.1, como uma memória de longo prazo declarativa, que também é chamada de explícita.

Memórias de longo prazo são informações que a mente pode armazenar por longos períodos de tempo (tipicamente acima de 1 minuto) e potencialmente durante toda a vida de uma pessoa. Essas memórias se subdividem em memórias declarativas e não-declarativas. Memórias declarativas são aquelas que podemos recuperar e formular conscientemente, na forma de uma declaração, como por exemplo uma equação matemática ou uma cena de filme. Em contrapartida, as memórias não declarativas são in-

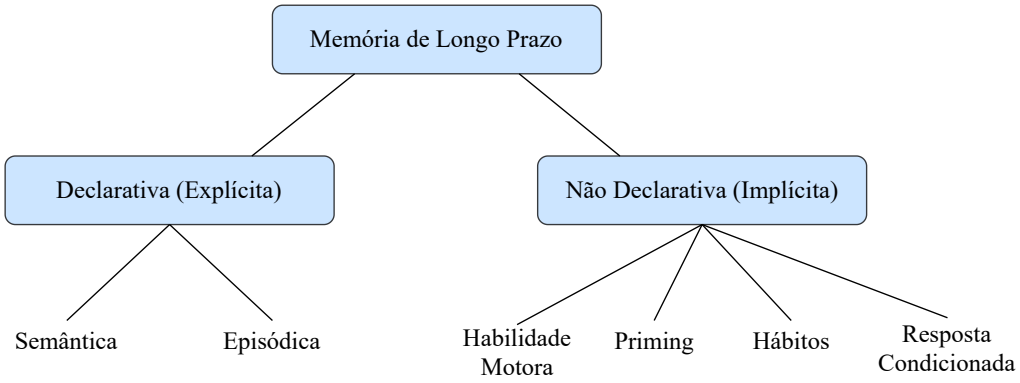


Figura 2.1 – Taxonomia das Memórias de Longo Prazo segundo definição de Squire e Zola-Morgan (1988).

conscientes, acessadas automaticamente, e não podemos elaborar com precisão qual é a informação nelas armazenada. Exemplos de memórias não declarativas incluem, por exemplo hábitos motores (como andar de bicicleta) que podem ser descritos em alto nível, mas que envolvem um mecanismo detalhado de como se equilibrar e aplicar força no pedal, que não poderia ser descrito de maneira declarativa. A memória declarativa pode ainda ser dividida entre memória episódica e memória semântica. Apesar de ambas poderem ser representadas por sentenças (declarações) em uma linguagem, a memória episódica distingue-se da memória semântica quanto ao conteúdo armazenado. Em termos gerais, as memórias semânticas contém fatos gerais sobre nós mesmos e o mundo, e as memórias episódicas armazenam eventos específicos que vivenciamos.

Tulving (1972) propôs que as memórias episódicas contêm representações conjuntas para: *o que*, *onde* e *quando* ocorreu um evento (conhecidas como informações WWW, do inglês *what, where, when*). Além disso, ele acrescenta que o processo de recordação de uma memória episódica acompanha uma experiência consciente do eu e sua projeção no tempo para experienciar eventos passados vividos ou eventos planejados no futuro (TULVING, 1985).

As memórias episódicas são essenciais para permitir o pensamento humano e são apoiadas por diversos mecanismos cerebrais e múltiplos processos. É importante considerar, além do mecanismo que define como a memória é recordada, também todas as etapas da experiência até a recordação. O sistema de memória episódica interage e influencia outros processos mentais, como a percepção, a memória de trabalho, a memória semântica, o raciocínio, etc., com o objetivo de atribuir sentido à nossa experiência e mantê-la na memória (BERNECKER, 2017).

2.1.1 Conceitos Básicos

Uma das maneiras pelas quais nosso cérebro entende o mundo é segmentando a experiência temporal contínua em unidades (GüLER *et al.*, 2023). Com isso, são construídas unidades de informações organizadas com as quais podemos perceber, interpretar e prever o mundo (EZZYAT; DAVACHI, 2011). Os **eventos** sensoriais detectados podem ser integrados a muitos estados mentais internos, como emoções, objetivos e motivações, para formar **episódios** complexos que nos dão uma narrativa de nossas vidas. Essa definição do conteúdo presente nos episódios é mais abrangente e coerente com estudos da memória episódica do que a definição original de Tulving (TULVING, 1972) sobre informações WWW. O psicólogo Conway (2005) argumenta em suas propostas de memória autobiográfica que, a informação da memória episódica fornece a base para organização do modelo de identidade de uma pessoa e uma representação rica de episódios com informações sensoriais, perceptuais, conceptuais e afetivas fornecem maior subsídio para identificação de uma memória como sendo uma experiência pessoal e não um fato histórico, plano, sonho, ou outras construções mentais.

As informações dos episódios percebidos são armazenadas e podem ser acessadas, mesmo que às vezes com dificuldade e de forma incorreta, em momentos futuros (WERNING, 2020). Além disso, as memórias episódicas também fornecem a base para a construção de outros pensamentos episódicos (MAHR, 2020). Episódios passados servem de estrutura para a construção de pensamentos episódicos futuros, nos quais podemos imaginar ou planejar nossas ações no futuro, bem como suas consequências e alternativas. O mesmo pode ser feito com experiências passadas, em que confabulações episódicas podem ser criadas imaginando-se o resultado de um conjunto diferente de ações tomadas no passado, levando até mesmo a oportunidades de aprendizado (MICHAELIAN; SUTTON, 2017). A capacidade cognitiva de se projetar em episódios reais ou imaginários no passado ou no futuro é conhecida como **Viagem Mental no Tempo**, sendo primeiro introduzida pelos psicólogos Suddendorf e Corballis (1997).

Memórias episódicas também são epistemicamente generativas, ou seja, são capazes de fornecer novos conhecimentos bases para crenças (MAHR; CSIBRA, 2018; WERNING, 2020). Por exemplo, depois de sair de casa, seu colega de quarto pode perguntar se o fogão estava desligado. Talvez você não tenha essa crença em mente e precise acessar sua memória episódica dos eventos do dia para se lembrar da sua ação de desligar

o fogão. Isso, por sua vez, criará e servirá de justificativa para a nova crença de que o fogão está desligado no momento. Um conhecimento útil a ser extraído das memórias episódicas é o progresso das metas de longo e curto prazo (CONWAY *et al.*, 2019). Logo, não é necessário manter o controle contínuo de várias metas com diferentes períodos de tempo e, em vez disso, basta computar seu progresso com base na memória episódica quando a meta se torna ativa.

Todas essas capacidades demonstram que, como muitos outros presentes no cérebro, o sistema de memória episódica constitui um conjunto complexo e interconectado de componentes e funcionalidades que ainda não são completamente compreendidos. Assim, os diferentes conjuntos de propriedades propostas na literatura para descrever modelos de memória não conseguem incorporar todas as evidências psicológicas e neurológicas encontradas nos diversos estudos realizados até o momento. Entretanto, é comum encontrar nesses modelos três processos principais realizados pelo cérebro que formam o sistema de memória episódica: codificação, armazenamento e recuperação (MICHAELIAN, 2016; KAHANA, 2020).

2.1.1.1 Codificação

Durante o processo de codificação, a experiência contínua é dividida em unidades de tempo discreto e integrada em uma única representação mental do episódio vivenciado, com a qual podemos identificar e interpretar as relações espaço-temporais do ambiente. A segmentação em eventos ocorre por meio da identificação de mudanças relevantes no estado externo (decorrentes da percepção das propriedades do ambiente e dos objetos que o compõem) ou do estado mental interno (como objetivo, emoção, carga cognitiva, entre outros), que identificam as bordas do evento (final de um evento e início de outro).

Para a segmentação de eventos a partir de mudanças externas observadas, a **Teoria da Segmentação de Eventos**, tendo o psicólogo Jeffrey M. Zacks como principal defensor, propõe que um modelo de evento atual seja construído na mente a partir das informações de entrada, enquanto as informações futuras são previstas (a partir de esquemas ou conhecimento geral). Quando o erro de previsão se torna alto, uma borda de evento é percebida e o modelo de evento é atualizado (ZACKS, 2020).

Por outro lado, a proposta de **Estabilidade Contextual** dos psicólogos e neurologistas Clewett e Davachi (2017) fornece uma melhor explicação para segmentação

de eventos a partir de mudanças internas à mente. Essa teoria propõe que o padrão de ativação neuronal forma um contexto associado às informações do evento e pode ser posteriormente re-instanciado. Mais importante, o contexto não é completamente estável e se desvia com o tempo, permitindo uma organização temporal da informação. Quando ocorre uma mudança no estímulo externo ou interno, a estabilidade contextual é alterada e diferenças mais acentuadas causam desvios mais fortes, indicando a segmentação do evento (CLEWETT; DAVACHI, 2017; CLEWETT *et al.*, 2019). Assim, quando mudamos de um cômodo para outro ou quando mudamos de objetivos (por exemplo, mudar o objetivo "prestar atenção na aula" para "comprar o almoço"), a assimilação dessas informações na mente causa uma variação na estabilidade contextual forte o suficiente para ser identificada como uma borda de evento e, consequentemente, faz com que a representação de cada evento seja divergente.

Vários estudos de detecção de ativação neuronal por ressonância magnética funcional ou eletrodos mostram como os tipos de mudança citados, como eventos inesperados ou mudança de objetivo, causam uma rápida mudança no padrão de ativação neuronal construído durante a experiência do evento, indicando a detecção de uma fronteira de evento que causou a atualização do modelo de evento. Além disso, no estudo neurocientífico de Baldassano *et al.* (2017) é apresentado evidências de que as alterações no padrão de ativação são encontradas hierarquicamente no cérebro. Conforme apresentado na Figura 2.2, as regiões responsáveis pelo processamento de informações sensoriais (detecção de cores, posição, movimento e assim por diante) detectam fronteiras em uma frequência mais alta do que as regiões de informações abstratas (identificação de objetos, faces, sons, entre outros), que, por sua vez, detectam fronteiras em uma frequência mais baixa do que as regiões cerebrais multimodais.

Isso aponta para uma organização hierárquica da representação de eventos no cérebro, em que contextos mais abstratos, como objetivos e narrativas, fornecem padrões estáveis que abrangem mudanças intermediárias de padrões, como mover-se entre salas ou encontrar novos objetos, o que, por sua vez, pode abranger mudanças ainda mais finas, como o movimento de objetos e a execução de ações motoras. Outros estudos também apresentam evidências que apontam para essa distinção de granularidade do evento, como a descoberta de células de fronteira e células de evento. Em outro estudo neurocientífico de Zheng *et al.* (2022), foram monitoradas as atividades individuais de um conjunto de

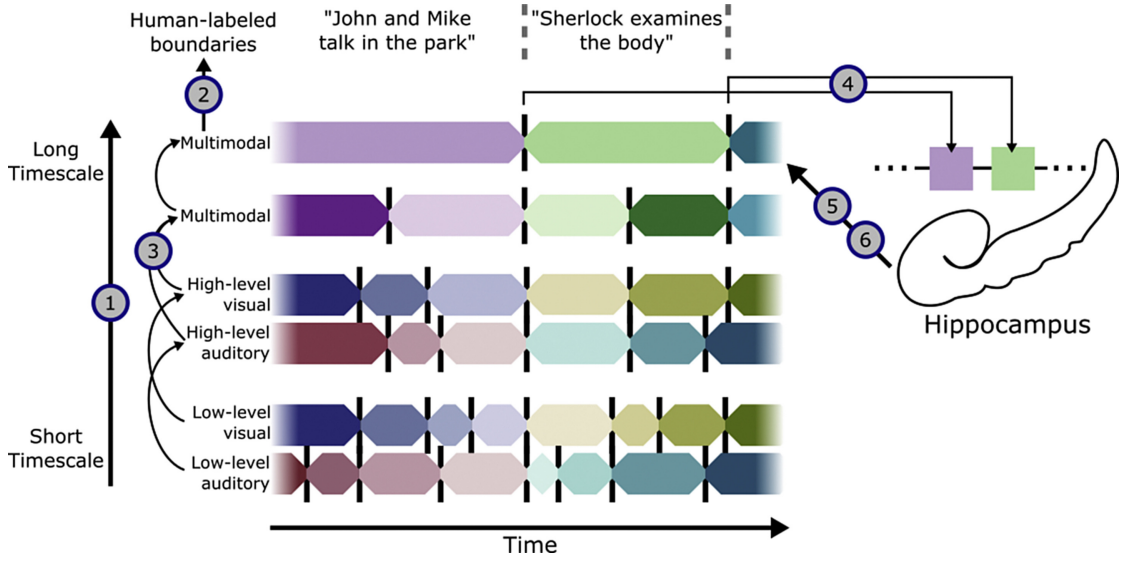


Figura 2.2 – Ilustração da Teoria da Segmentação de Eventos para um exemplo de observação de duas cenas de um filme - extraído de Baldassano *et al.* (2017).

neurônios do lobo temporal medial de pacientes enquanto assistiam vídeos selecionados. Os vídeos continham três tipos de transição: nenhuma transição, na qual a sequência de imagens do vídeo é contínua; transição fraca, na qual a cena exibida pelo vídeo é a mesma, porém há uma troca do ângulo da câmera; transição forte, onde a cena sendo exibida é substituída por outra cena. Foram identificados dois grupos distintos de neurônios com padrão de ativação dependente do tipo de transição. O primeiro grupo, nomeados células de fronteira, possuem aumento significativo na atividade quando uma transição fraca ou forte ocorre, enquanto o segundo grupo, nomeado células de evento, possuem aumento de atividade somente quando transições fortes ocorrem.

O padrão de ativação dos neurônios não é correlacionado com a mudança de cor ou luminosidade que ocorre nas transições utilizadas. Porém, é observado que erros de predição (transições menos esperadas) são altamente correlacionados com a atividade desses neurônios, indicando assim uma dependência do contexto sendo experienciado. Além disso, a presença de células com respostas distintas a transições fracas e fortes reforça a teoria da estrutura hierárquica de codificação de eventos (ZHENG *et al.*, 2022).

2.1.1.2 Armazenamento

A informação episódica armazenada é uma representação parcial da representação criada pela codificação do evento percebido. Em geral, o termo **traço de memória** é usado para diferenciar entre as informações armazenadas na mente e a representação de um episódio lembrado (memória episódica). Originalmente, Tulving propôs que traços de

memória episódica são resumos de experiências contendo informações WWW (TULVING, 1972). Esse modelo apresenta falhas por não tratar do limite de capacidade da memória humana e a reativação de informações sensoriais na mente durante a recuperação de memórias (WHEELER *et al.*, 2000), porém continua sendo amplamente usado em modelos computacionais devido a sua simplicidade de implementação e uso.

Atualmente, a teoria da **memória construtiva** tem sido defendida por pesquisadores nas áreas de psicologia e neurociência (MICHAELIAN, 2016; ADDIS, 2018; CHENG *et al.*, 2016; WERNING, 2020). Essa teoria defende que o processo de recordação de um episódio envolve a construção de uma representação mental a partir de fragmentos de memória episódica, bem como de outras formas de conhecimento (como memórias perceptuais e semânticas), sendo que diferentes pesquisadores defendem diferentes proporções entre informação construída em relação a informações diretamente recuperadas (CHENG *et al.*, 2016; WERNING, 2020). Dessa forma, após a codificação do episódio é armazenada a essência do mesmo, em que apenas parte da representação do episódio experienciado é transferida para a memória e, a partir dessa essência do episódio, a mente pode reconstruir, com certo nível de confiabilidade, as informações sobre o episódio (WERNING, 2020). Ou seja, os episódios não são armazenados por completo na memória como uma cópia instantânea do estado atual, mas somente uma representação parcial ou abstrata do mesmo que permite a reconstrução verossímil do episódio original.

2.1.1.3 Recuperação

A recuperação de memórias é vista como um processo de busca de eventos vivenciados no passado com alta correlação com uma informação parcial atual, a **pista** (KAHANA, 2020). Esse processo pode ocorrer conscientemente, quando é feito um esforço deliberado para buscar um evento passado, ou automaticamente, quando as informações de um ou mais eventos passados são trazidas à consciência devido à alta associatividade entre itens de eventos codificados atualmente e eventos passados. Além disso, de maneira mais evidente no processo de recordação consciente, a recuperação de um evento específico a partir de uma pista inicial pode envolver um processo recursivo. Assim, quando uma pista inicial com poucas informações leva à recordação de um evento, uma nova pista é extraída da memória recordada para alimentar novamente o processo de recuperação (DINGS; NEWEN, 2021). Por exemplo, ao tentar lembrar onde deixou as chaves, você primeiro lembra que tinha as chaves para destrancar a porta quando chegou em casa com

as compras, o que o leva a lembrar que foi até a cozinha para guardar as compras, o que o faz lembrar que deixou as chaves no balcão da cozinha.

O fator importante da memória recuperada, como apontado pela teoria da **memória construtiva**, é que a representação mental do evento lembrado não é a mesma do evento vivenciado (CHENG *et al.*, 2016; MICHAELIAN; SUTTON, 2017). O processo de recordação envolve a reconstrução, também chamada de simulação, do evento passado, em que as informações de diferentes traços (eventos semelhantes ou sequenciais) e tipos (semântico, perceptual, autobiográfico, etc.) de memória são ativadas e integradas em uma representação verossímil da experiência passada.

2.2 Arquiteturas Cognitivas

O estudo de arquiteturas cognitivas é uma subárea da ciência cognitiva voltada para o desenvolvimento de modelos teóricos e computacionais de processos cognitivos. A definição de uma arquitetura envolve a descrição de componentes de processamento e armazenamento de informações em uma estrutura que capacite um agente inteligente a representar, adquirir e utilizar conhecimento para perseguir objetivos (LAIRD, 2012). De maneira geral, agentes inteligentes são sistemas situados em um ambiente e capazes de interagir com o mesmo (modificando-o e percebendo suas mudanças), buscando atingir um estado objetivo (FRANKLIN; GRAESSER, 1997). Uma arquitetura cognitiva define uma estrutura, que guia o desenvolvimento de aplicações específicas, mas o conhecimento presente nas memórias do agente é mutável, podendo ser adaptado para a tarefa a ser cumprida.

As arquiteturas cognitivas surgiram como ramificação da ciência cognitiva durante os anos 80 a partir dos avanços teóricos da área nos anos anteriores (NEWELL; SIMON, 1961; SIMON; NEWELL, 1971). Assim, os modelos desenvolvidos comumente dividem o processo de geração de comportamento dos agentes em capacidades cognitivas estudadas nas áreas da neurociência, psicologia, antropologia e linguística, como percepção, atenção, memória, aprendizagem, aquisição de linguagem, comunicação, entre outros (LAIRD, 2012). Diversas arquiteturas cognitivas foram propostas desde o surgimento da área, buscando trazer avanços no entendimento e desenvolvimento dos diferentes processos cognitivos necessários e suas interações para geração de comportamento semelhante ao humano.

O desenvolvimento de arquiteturas cognitivas computacionais têm como foco a criação de sistemas com capacidades de resolução de problemas em diferentes domínios. Um estudo realizado por Kotseruba e Tsotsos (2020) analisou 84 arquiteturas cognitivas diferentes filtradas de uma lista inicial de 195 arquiteturas. Os autores apontam que o objetivo final do estudo em arquiteturas cognitivas é a modelagem da mente humana e a criação de sistemas inteligentes com comportamento humano. Devido a participação da memória episódica em diferentes processos cognitivos e sua importância para geração dos comportamentos humanos, diversas propostas para módulos computacionais de memória episódica em arquiteturas cognitivas foram feitas. Entretanto, diversos problemas que requerem estudos mais aprofundados foram apontados pelos autores. Dentre eles, a exploração superficial de componentes da memória episódica nas arquiteturas. Além disso, é destacado o uso predominante de memórias episódicas que somente armazenam cópias do estado da mente com um *timestamp* e a necessidade de explorar outras representações mais elaboradas para permitir agentes com maior escopo temporal. Na próxima seção, destacamos algumas das propostas feitas de modelos computacionais para inclusão de sistemas de memória episódica em arquiteturas cognitivas.

2.3 Trabalhos Relacionados

A arquitetura SOAR[1], desenvolvida por Laird, Rosenbloom e Newell (LAIRD *et al.*, 1987), utiliza um módulo de memória episódica inspirado nas definições de Tulving (1972). A memória episódica da arquitetura SOAR inclui processos de codificação, armazenamento e recuperação, conforme ilustrado na Figura 2.3. A codificação de um episódio é iniciada automaticamente sempre que o agente executa uma ação externa. A representação da memória é construída pela cópia dos elementos mais relevantes presentes no estado atual da memória de trabalho. Entretanto, os elementos dos subestados (utilizados nos subprocessos para tomada de decisões) e as memórias episódicas recuperadas atualmente na memória de trabalho não são armazenados. Além disso, uma anotação temporal do instante de armazenamento é adicionada à instância de memória gerada. Dessa forma, o episódio armazenado é somente uma cópia das estruturas de dados de parte da mente do agente no momento da ação, mas sem representar o processo de raciocínio e os estados anteriores de percepção que levaram ao estado atual.

[1] State, Operator And Result

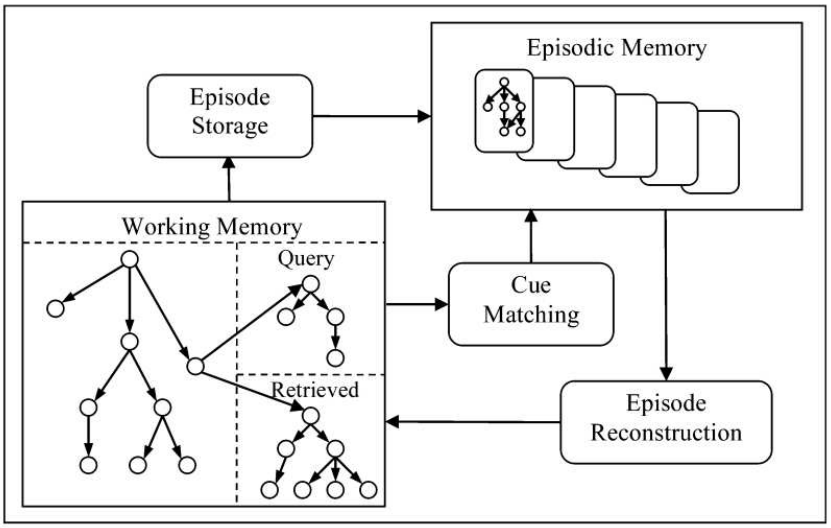


Figura 2.3 – Módulos utilizados para processamento de memórias episódicas na arquitetura SOAR - extraído de Laird (2012).

As memórias são recuperadas pela comparação de uma pista, formada por uma estrutura parcial de elementos da memória de trabalho, com as memórias episódicas armazenadas. O episódio ativado é inserido na memória de trabalho juntamente com informações sobre seu grau de similaridade com a pista, permitindo que o agente determine sua utilidade. Adicionalmente, depois que um episódio é recuperado, o agente pode optar por recuperar episódios anteriores ou posteriores de forma sequencial.

O modelo de memória episódica do SOAR concentra-se em uma representação de estados e não captura informações sobre eventos que ocorrem entre estados, sendo um dos problemas destacados no estudo de Kotseruba e Tsotsos (2020). Dessa forma, a representação armazenada é uma duplicação direta das informações do agente e não tem interpretações de alto nível.

Diversas outras arquiteturas cognitivas propõem sequências de representações simbólicas de estados para implementação de memórias episódicas. Em Dodd e Gutierrez (2005), a representação do estado do agente contém elementos simbólicos perceptuais, semânticos, afetivos, tarefas executadas e objetivo ativo. A segmentação temporal dos estados é realizada na conclusão de um objetivo. Semelhantemente, Kuppuswamy *et al.* (2006) agrega diferentes tipos de informação na representação de estado da memória episódica, porém permite flexibilidade na escolha desses tipos a depender do domínio de aplicação. A segmentação dos estados pode ser realizada em intervalos de tempo fixos ou na satisfação de estados internos (e.g. fome). Em Tecuci e Porter (2009) o conteúdo dos

episódios são divididos em três dimensões (contexto, conteúdo e resultados) e indexados em dois níveis. No primeiro nível é indexada cada dimensão individualmente e no segundo nível a estrutura geral do episódio. Esse modelo é considerado isolado do sistema ou agente gerando as informações dos episódios, logo não considera como os mesmos são segmentados.

A arquitetura ICARUS (LANGLEY *et al.*, 1989) concentra-se na cognição incorporada e usa um sistema simbólico baseado na percepção, onde estruturas hierárquicas são construídas a partir da percepção para formar crenças, objetivos, categorias e outras informações na memória de trabalho e de longo prazo do agente. As memórias de longo prazo da arquitetura são estruturadas hierarquicamente, criando árvores que armazenam as informações de categorias, habilidades, objetivos e episódios.

O modelo de memória episódica da arquitetura ICARUS adota uma representação híbrida de episódios genéricos (esquemas) e específicos organizados em uma árvore (MéNAGER *et al.*, 2022). Conforme apresentado na Figura 2.4, os nós das folhas correspondem a eventos específicos e os nós intermediários são esquemas que generalizam os nós filhos. Um episódio é composto pela estrutura em grafo contendo os objetos identificados conectados à percepção do agente que os fundamenta, bem como as crenças derivadas deles com base em categorias de relações. Assim, ao agregar episódios em um esquema, é contruída uma rede bayesiana em que cada nó armazena as distribuições de probabilidade de seus possíveis valores juntamente com os valores dos nós pais. Quando um novo episódio a ser incorporado à memória de longo prazo é apresentado, ele é comparado com os nós presentes na estrutura em árvore da memória, recursivamente, do nó raiz até as folhas. Quando um nó é encontrado com um esquema cuja correspondência com o novo episódio é maior do que a correspondência de seus nós filhos, o novo episódio é adicionado como um nó filho e os esquemas presentes no caminho até o nó raiz, incluindo o nó raiz, são atualizados. Durante a recuperação de uma memória, uma pista, formada por um gráfico de dependência parcial, é comparada com a estrutura em árvore da memória de longo prazo, semelhante ao processo de incorporação de novos episódios. Quando o nó com a melhor correspondência é um episódio específico (nó folha), ele é retornado como resultado da pesquisa. Se for um esquema, é feita uma inferência a partir da rede bayesiana descrita pelo esquema e condicionada pelos valores das pistas, para instanciar o episódio mais provável.

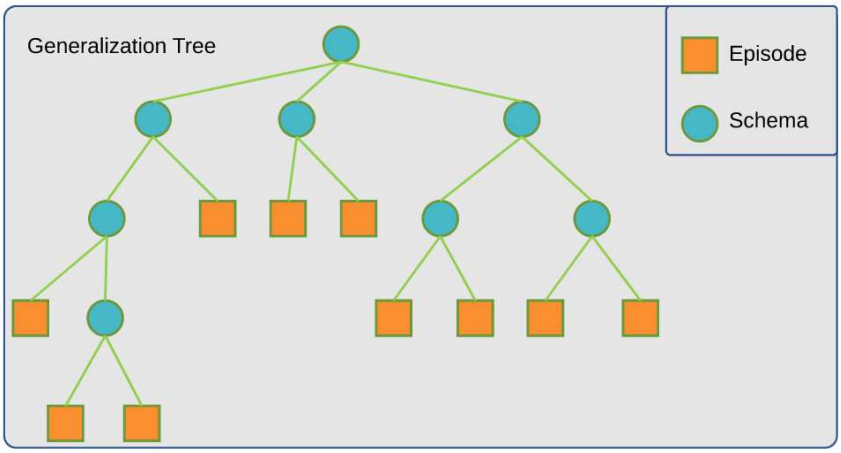


Figura 2.4 – Esquema de representação de memórias episódicas da arquitetura ICARUS - extraído de Ménager *et al.* (2022).

Assim como em ICARUS, outros modelos com formas de organização hierárquica das memórias episódicas foram propostos. Em Brom *et al.* (2007), uma estrutura hierárquica de tarefas que podem ser executadas é mantida pelo agente. Episódios são estados simbólicos de informações perceptuais e afetivas do agente e são organizados hierarquicamente com base na tarefa executada no momento de armazenamento. Dessa forma, a estrutura hierárquica dos episódios armazenados segue a estrutura hierárquica das tarefas conhecidas pelo agente. Deutsch *et al.* (2008) utiliza *templates* para detecção de estados e sequências de eventos a partir da informação perceptual do agente. Para formação de um estado, *templates* de imagens, estados afetivos e ações são comparados com a informação perceptual e valores de probabilidade de *match* são atribuídos a cada *template* para formar o estado atual. Os *templates* de maior saliência para cada feature (imagem, emoção e ação) formam o evento atual. A detecção de episódios utiliza diagramas de transição de estados para identificar eventos pertencentes a instâncias de episódios e os armazenam como sequências de eventos. Com isso, um mesmo evento pode ser relevante para múltiplos episódios, criando uma memória episódica com sobreposições.

A arquitetura Cuâyôllôtl[2] busca imbuir um agente ou sistema com comportamento semelhante ao humano (CORCHADO *et al.*, 2021). Os diversos módulos e capacidades cognitivas da arquitetura são fortemente inspirados nos estudos neurocientíficos e psicológicos da mente humana (MARTIN *et al.*, 2022). O módulo de memória episódica da Cuâyôllôtl apresentado na Figura 2.5 contém diversos processos para criação, armazenamento e recuperação de contextos espaciais e valores afetivos de cenas. O módulo recebe

[2] Cérebro em Nahuatl, uma língua nativa mexicana

um fluxo de imagens como entrada e cada frame é codificado em uma representação simbólica da distribuição espacial 2D dos objetos detectados. Essa representação carrega a categoria e posição de cada objeto no plano da imagem de entrada. Paralelamente, processos de avaliação afetiva atribuem valores afetivos a cada objeto identificado do frame e os agrega em um valor afetivo geral. Esse valor afetivo é então utilizado para reforçar associações entre objetos e adicionado à representação simbólica do frame. O armazenamento de novos frames ocorre sempre que não há nenhum frame semelhante já presente na memória. Caso haja, o valor de ativação da memória armazenada é incrementado. Por fim, associações são criadas ou fortalecidas entre sequências de frames armazenados (MARTIN *et al.*, 2021; MARTIN *et al.*, 2022).

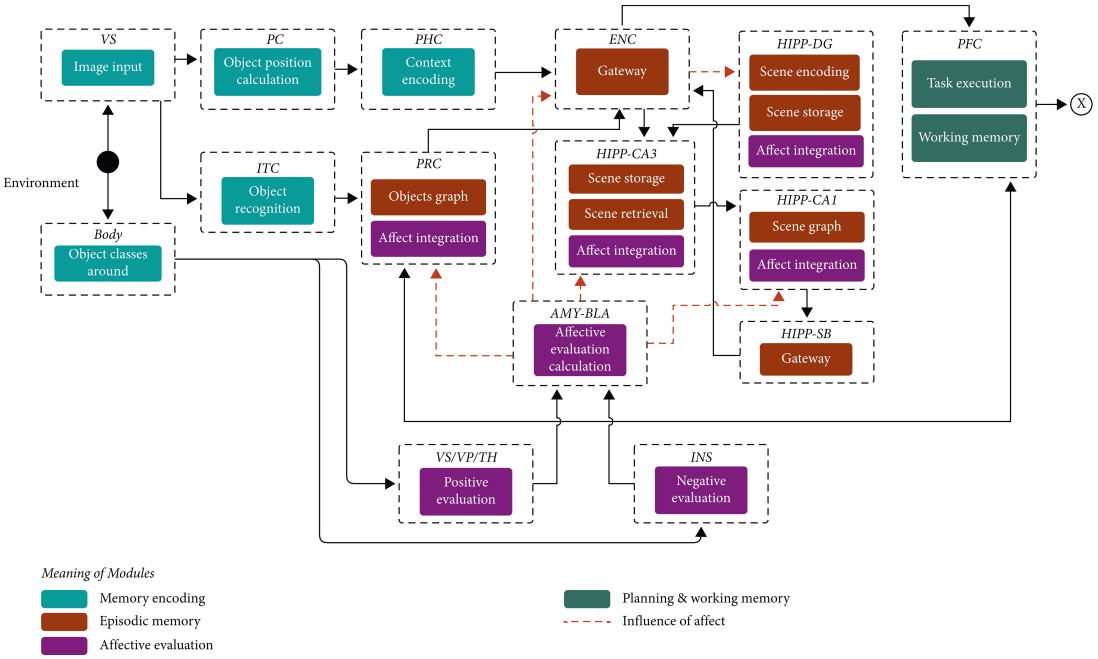


Figura 2.5 – Sistema episódico da arquitetura Cuâyôllôtl - extraído de Martin *et al.* (2021)

O modelo de memória episódica da Cuâyôllôtl gera relações entre estados simbólicos que representam contextos espaciais e valores afetivos. Embora esse modelo seja baseado em estados simbólicos, o mesmo apresenta o diferencial de considerar um mecanismo integrado para segmentação de estados pelo modelo com base na diferença do contexto espacial ao longo do tempo, ao invés de depender de segmentações pré-determinadas de eventos.

Semelhante ao sistema episódico da Cuâyôllôtl, o framework EM-SOL (ZOU *et al.*, 2021) propõe um modelo de memória episódica com segmentação integrada de eventos baseada na variação de *SIFT features* extraídas das imagens do ambiente. Adicionalmente,

os eventos são agrupados em episódios quando o agente alcança o local objetivo destino, apresentando assim dois níveis hierárquicos de segmentação temporal de estados.

Por fim, algumas arquiteturas propõem modelos de memória episódica baseados em diferentes teorias e evidências da psicologia e neurociência, porém ainda estão em desenvolvimento e não possuem estudos específicos sobre sua implementação, como os casos das memórias transientes e declarativas do LIDA (FRANKLIN *et al.*, 2016), a memória episódica do CLARION 6.0 (SUN, 2003; WILSOM, 2013), MECA (GUDWIN *et al.*, 2017), e Haikonen (HAIKONEN; HAIKONEN, 2012).

2.4 Considerações Finais

A memória episódica é alvo de estudo em diferentes áreas como filosofia, psicologia e neurociência, mostrando-se um componente importante da inteligência humana e altamente complexo. Diferentes teorias são exploradas de forma paralela e pontos de discordância existem entre pesquisadores. Entretanto, existe a convergência de alguns aspectos.

A segmentação temporal da informação adquirida pela mente é necessária para realizarmos a interpretação do nosso mundo, porém o mecanismo de segmentação implementado pelo cérebro humano ainda não é completamente entendido. Por um lado, a Teoria de Segmentação de Eventos advoca por modelos mentais de eventos que, ao apresentarem erros de predição, indicam segmentações, enquanto a teoria de Estabilidade Contextual propõe que pertubações no padrão de ativação neuronal causadas pela assimilação de novas informações a mente causam a identificação de transições de eventos.

Durante o armazenamento, uma representação modificada do episódio é integrada a memória de longo-prazo, passando por um processo de extração de traço da memória em que parte da informação é abstraída. Essa informação deve ser suficiente para reconstrução do episódio quando o mesmo é relembrado, porém não se conhece o nível de abstração que ocorre no armazenamento de longo-prazo.

Nesse trabalho, adota-se uma abordagem conciliadora, na qual busca-se deixar espaço para o uso de diferentes teorias e possíveis combinações das mesmas. Assim, propõe-se um sistema de memória episódica cuja segmentação de eventos seja feita em dois níveis, um inspirado pela Teoria de Segmentação de Eventos e outro inspirado pela Estabilidade Contextual. Para isso incorporamos também a descoberta de células de fronteira presentes

no cérebro humano.

Com isso, pretende-se adereçar o problema de representação de episódios em estados, encontrada em arquiteturas cognitivas como o SOAR e ICARUS, com uma representação de eventos que incorpore informação da transformação ocorrida no período. Além disso, ao incorporar um mecanismo específico para segmentação temporal de eventos e episódios, endereçamos a falta ou simplicidade desses mecanismos nas arquiteturas exploradas. Por fim, é observado também que poucas arquiteturas utilizam processos de de extração de traço do episódio, sendo que muitas realizam a cópia direta das estruturas de dados do episódio codificado para a memória de longo-prazo e algumas realizam processos de abstração, como em ICARUS, ou de criação de conexões entre elementos de episódios, como em Cuâyôllôtl e EM-SOL. Dessa forma, o modelo de arquitetura cognitiva com sistema episódico proposto neste trabalho busca solucionar essas lacunas observadas.

3 CST e Ideas

Para a implementação da arquitetura cognitiva foi utilizado o *Cognitive Systems Toolkit* como principal ferramenta de desenvolvimento. Além disso, o modelo de representação de conhecimento *Ideas* foi adotado como padrão para modelagem de toda informação processada pela arquitetura. Embora os detalhes técnicos de operação e programação utilizando o CST e *Ideas* serão de grande relevância no Capítulo 5, já no Capítulo 4, onde é apresentado o modelo teórico do sistema de memória episódica proposto, alguns dos conceitos de construção de arquiteturas cognitivas com CST e modelagem de conhecimento com *Ideas* são utilizados.

Dessa forma, nesse capítulo é apresentado os principais conceitos da ferramenta CST, introduzindo os elementos principais que o compõe, assim como o modelo de representação *Ideas*. É destacado também o componente *memory container* do CST e sua utilização para implementação de mecanismos de subsunção e motivação.

3.1 O *Cognitive Systems Toolkit*

O *Cognitive Systems Toolkit* (CST) é um toolkit de desenvolvimento e implementação de arquiteturas cognitivas criado e mantido por professores e pesquisadores da Universidade Estadual de Campinas (PARAENSE *et al.*, 2016). A concepção dos componentes básicos do CST foi inspirada em arquiteturas cognitivas existentes e estabelecidas na área, especialmente o Clarion (SUN, 2005) e LIDA (BAARS; FRANKLIN, 2009). Como um *toolkit*, o CST tem o objetivo de fornecer ferramentas flexíveis para a implementação de diversas arquiteturas cognitivas. Dessa forma, o desenvolvimento de uma arquitetura cognitiva com o CST pode incorporar diferentes modelos de processos cognitivos, sem se restringir às implementações fixas existentes de arquiteturas cognitivas propostas na literatura.

O CST possui dois elementos como base para desenvolvimento: *codelets* e *memory objects*. Na implementação de uma arquitetura cognitiva com o CST, o comportamento e função de diversos desses elementos podem ser definidos e organizados de acordo com os processos mentais escolhidos para compor a arquitetura. A Figura 3.1 apresenta os componentes bases do CST e seu princípio de organização em uma arquitetura.

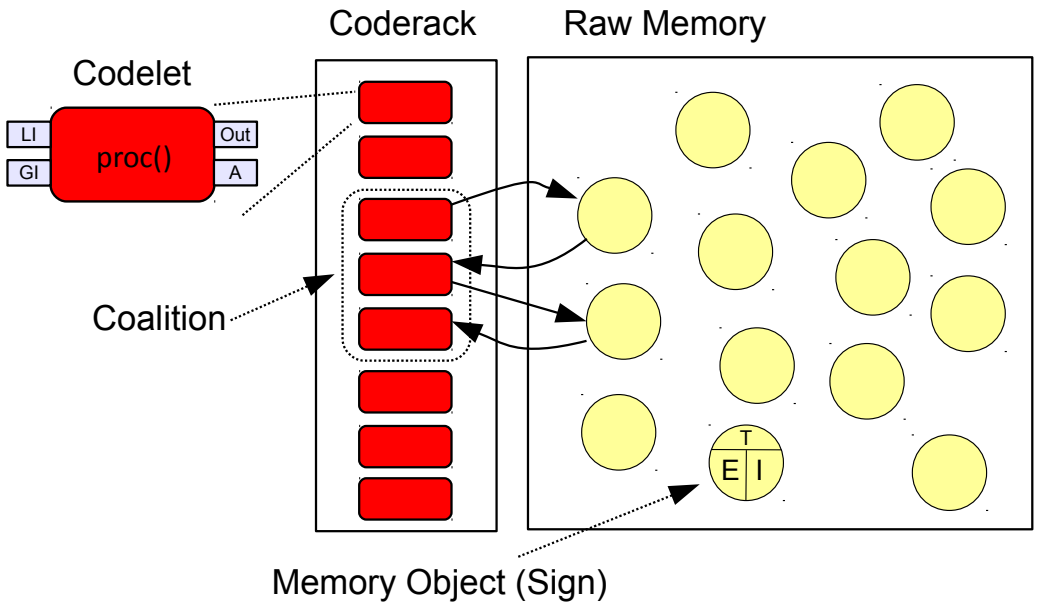


Figura 3.1 – Componentes bases do *Cognitive Systems Toolkit* - extraído de (PARAENSE *et al.*, 2016).

Codelets são pedaços de código mínimos e não bloqueantes que executam uma tarefa simples de forma contínua e cíclica. Em uma arquitetura construída com o CST, todo processamento das capacidades cognitivas é realizado por *codelets*, dessa forma a arquitetura é conceitualmente um sistema multi-agentes assíncrono e paralelo. *Memory objects* são armazenadores genéricos de informações e guardam qualquer informação auxiliar ou permanente necessária para operação dos *codelets*. As entradas e saídas dos *codelets* são comunicadas por *memory objects*.

3.1.1 Memory Container, Subsunção e Impulsos

Em Gudwin *et al.* (2017) é introduzido o elemento *memory container* ao CST. Um *memory container* é um tipo especial de *memory object* que armazena a informação de cada fonte de entrada separadamente e contém um mecanismo de decisão para definir a informação de saída. Conforme ilustrado na Figura 3.2, o *memory container* armazena a informação de cada entrada em *memory objects* distintos juntamente com um valor de avaliação. O valor de avaliação é fornecido pelo *codelet* que está inserindo a informação no *memory container* e representa o nível de importância do mesmo. O processo de escolha do *memory container* simplesmente escolhe a informação com maior nível de avaliação e a transmite para saída. A saída opera como um *memory object*, cuja informação pode ser acessada por qualquer *codelet* com entrada conectada ao *memory container*.

O componente *memory container* foi introduzido para facilitar a implementa-

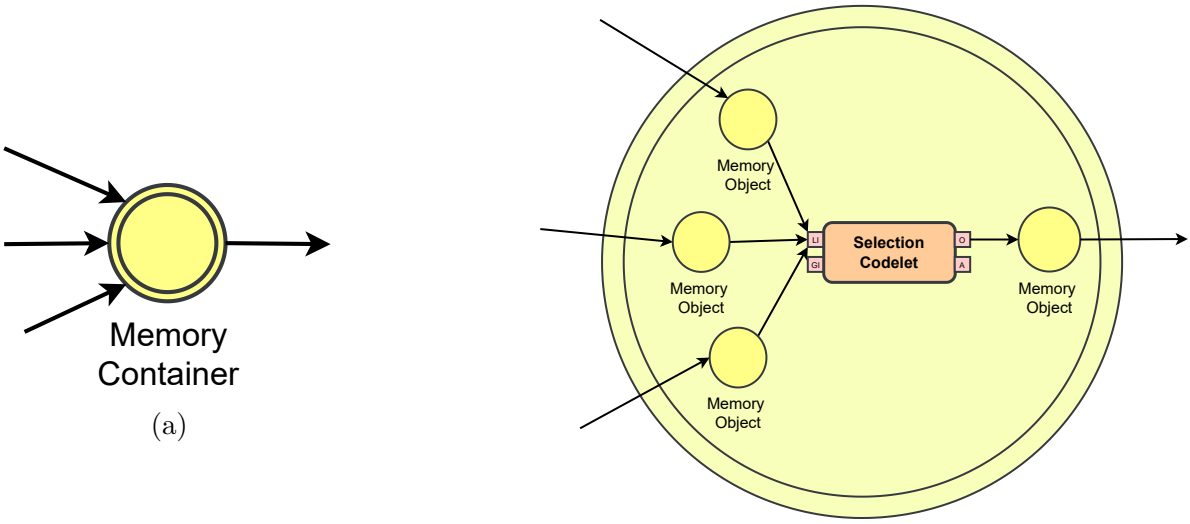


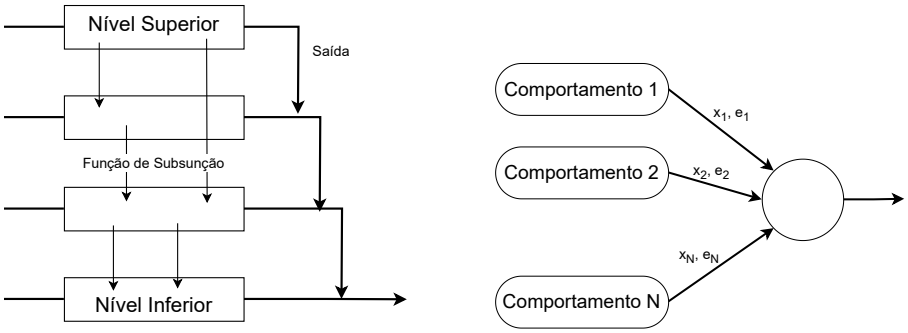
Figura 3.2 – Elemento *Memory Container*. (a) Representação de um *memory container* conforme padrão esquemático do CST; (b) Representação do funcionamento interno de um *memory container* - adaptado de Gudwin *et al.* (2017).

ção de mecanismos de subsunção dinâmica. O modelo de sistema de subsunção, introduzido por Brooks (1986), propõe uma reformulação na organização dos processos realizados por um agente[1]. Até então, a abordagem mais comum era a decomposição do sistema em módulos para processamento em série da informação e geração de comportamento. Brooks aponta que esse modelo de desenvolvimento de arquiteturas apresenta grande problemas de escalabilidade, onde a complexidade na inclusão de novas capacidades cresce com o tamanho do sistema. Dessa forma, Brooks (1986) propõe arquiteturas em paralelo, em que comportamentos são modelados individualmente e então agregados em um sistema de subsunção.

No esquema de subsunção clássico, os comportamentos são organizados em níveis, onde comportamentos de níveis mais altos possuem precedência sobre a execução de comportamentos de níveis inferiores. Entretanto, isso apresenta a desvantagem de uma hierarquia fixa, na qual a prioridade dos comportamentos não pode ser alterada durante a execução do sistema. Assim, o mecanismo de subsunção dinâmica apresenta uma solução a este problema, onde a prioridade de cada comportamento é definida de forma dinâmica. A Figura 3.3 compara o princípio de implementação da subsunção clássica, onde a prioridade dos comportamentos é fixa, e a subsunção dinâmica, onde cada comportamento gera um nível de avaliação e_i juntamente da informação de comportamento x_i e então a

[1] A proposta original em Brooks (1986) era voltada para arquiteturas aplicadas ao controle de robôs.

saída é selecionada com base no maior valor de avaliação. Fica claro aqui, como o *memory container* é, então, um componente do CST voltado para implementação de subsunção dinâmica em arquiteturas cognitivas.



(a) (b)

Figura 3.3 – Representação comparativa dos modelos de (a) subsunção clássica e (b) subsunção dinâmica - adaptado de (GUDWIN *et al.*, 2017)

Em específico, neste trabalho o *memory container* é utilizado na implementação do módulo motivacional do agente, o qual é responsável pela seleção de objetivos do agente. A definição do objetivo de um agente é um processo integral dentro de uma arquitetura cognitiva. Dentre os métodos de escolha de objetivos, o uso de um sistema motivacional atribui necessidades a um agente para guiar essa escolha.

Sistemas motivacionais são responsáveis por determinar a forma que um agente escolhe um objetivo, causando um comportamento. As motivações de um agente geram comportamentos a serem realizados com base nas percepções do agente e suas necessidades. Tradicionalmente, sistemas motivacionais de arquiteturas cognitivas se baseiam na teoria motivacional de Hull (1943), que considera a existência de necessidades internas que nos motivam a realizar comportamentos com o objetivo de satisfazê-las (GUDWIN, 2019). Cada necessidade pode ser associada a um valor de satisfação[2] (denominado *drive*) que aciona comportamentos específicos para satisfazer as necessidades. Dessa forma, arquiteturas cognitivas baseadas na teoria motivacional de *drives* possuem módulos que avaliam a satisfação de um conjunto de necessidades para geração de *drives*. Com isso, um processo de decisão pode escolher qual *drive* deve ser satisfeito no momento e qual comportamento deve ser gerado para cumprir esse objetivo.

Além dos *drives*, Baumeister (2016) caracteriza uma segunda forma de moti-

[2] Na verdade, de insatisfação: quanto maior a insatisfação, maior será o *drive*.

vação baseada em impulsos. Impulsos são gerados em instantes específicos e correspondem à motivação de realizar um comportamento específico naquele momento. Diferentemente de *drives* que são avaliados constantemente, mesmo enquanto comportamentos para satisfazê-los estão em execução, os impulsos são gerados em um certo instante de tempo e desaparecem quando são satisfeitos ou se tornam irrelevantes.

O surgimento dos impulsos depende da interação com o ambiente e do estado interno do agente. Situações oportunas para a satisfação de uma necessidade, mesmo que possua baixo *drive*, podem gerar impulsos para execução de um comportamento específico naquele instante. Além disso, mudanças no estado interno do agente podem despertar um impulso para satisfação de uma necessidade, mesmo sem uma situação oportuna ter sido apresentada. Por exemplo, durante o trajeto até o trabalho encontramos uma nova cafeteria, um impulso para consumir algum alimento pode surgir, mesmo quando não estamos com fome. Em contrapartida, após muito tempo sem comer uma pizza, podemos ter um impulso de comer pizza, mesmo sem estar com fome ou uma situação oportuna de realizar tal comportamento tenha sido apresentada (BAUMEISTER, 2016; GUDWIN, 2019).

3.2 *Computational Ideas*

Em Camargo *et al.* (2022) foi originalmente introduzido o conceito de *Ideias Computacionais* ou, como no original, simplesmente *Ideas*, que foi posteriormente incorporado ao CST como uma forma padrão de representação de conhecimento de diferentes tipos. O termo *Idea* é adotado dos trabalhos do filósofo John Locke, o qual, em termos simples, apresenta as *Ideas* como toda representação de entidades do mundo que surge na mente, sejam elas representações simples adquiridas diretamente dos nossos sensores ou complexas, formadas por composições de *Ideas* mais simples (LOCKE, 1847). Assim, *Computational Idea* é um modelo conceitual de representação do conhecimento que fornece blocos genéricos de representação, que podem ser organizados para formação de conceitos complexos.

Segundo o modelo das *Ideas*, a existência é um contínuo espaço-temporal no qual estão inseridos o agente e tudo que há no ambiente. Essa existência é representada por *ideas* de diferentes tipos, que podem corresponder a fragmentações ou composições de outras *ideas* (Figura 3.4). Em um primeiro contato da existência com os sensores, a

atenção volta-se à percepção de unidades de existência que se destacam como conjuntos coerentes durante a evolução dos valores de um conjunto de sensores. Partindo de uma visão focada em unidades de existência, denominadas objetos, a formulação das *ideas* propõe a hipótese de duas direções para identificação de outros tipos de *ideas*. Em uma abordagem *top-down* por decomposição, os objetos podem ser detalhados em propriedades, elicitando as diferentes características que compõe a unidade de existência. Já em uma abordagem *bottom-up*, ao incorporar a percepção do tempo, as variações dos objetos e a evolução de suas relações são compostas em episódios.

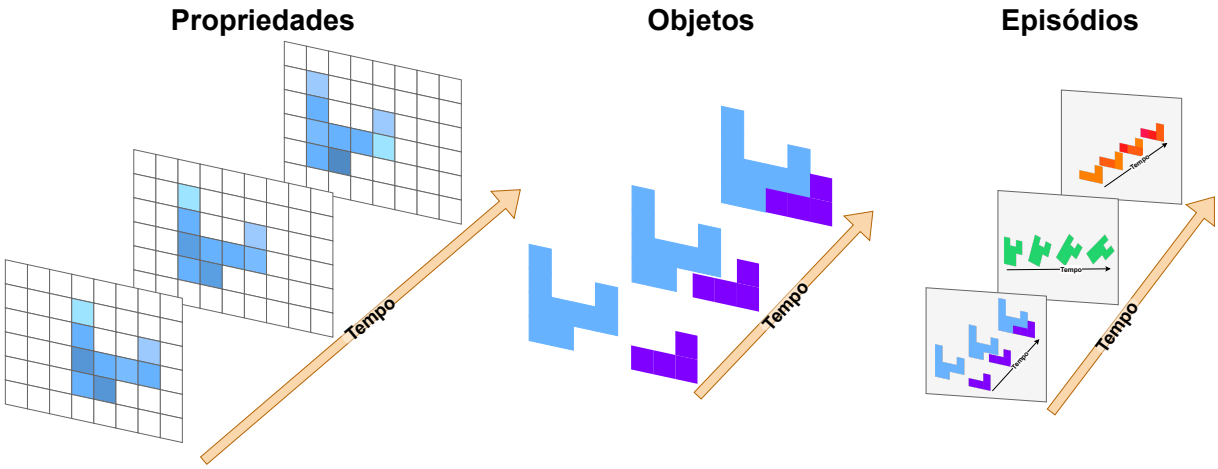


Figura 3.4 – Representação da existência em propriedades, objetos e episódios. Propriedades são adquiridas por conjuntos de sensores, cujas variações ao longo do tempo permitem a inferência de objetos que são organizados em episódios conforme a mudança de suas estruturas no espaço-tempo.

A implementação computacional em Java das *Ideas* para o CST é denominada *computational idea*, sendo representada pela classe *Idea* conforme o diagrama da Figura 3.5. Uma *Idea* possui três atributos principais que carregam sua informação: nome, valor e uma lista de *Ideas* associadas, que funcionam como índices para outras *Ideas* que integram a composição da *Idea* em questão. O nome é uma *string* que indica o símbolo pelo qual a *Idea* é referenciada. O valor pode ser um objeto Java qualquer, e contém o dado carregado pela *Idea*, como um número, *string*, vetor ou matriz. Por fim, uma lista de *Ideas* associadas agrega informações complementares para a *Idea* em questão. Dessa forma, pode-se compor diversas *Ideas* em uma informação mais complexa.

Adicionalmente, uma *Idea* possui ainda uma categoria e um escopo. A categoria indica o tipo a qual a informação da *Idea* se refere. Essa categoria é uma *string* podendo assumir os valores *"Episode"*, *"AbstractObject"* ou *"Property"*. Existem tam-

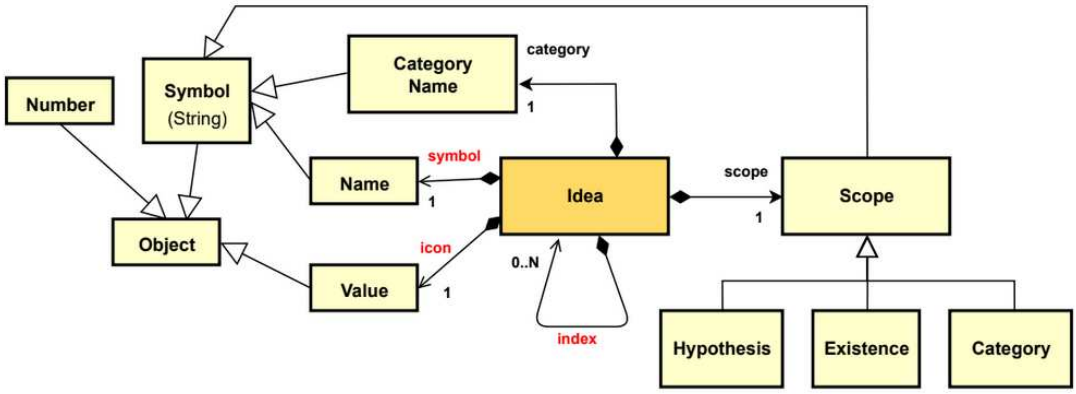


Figura 3.5 – Diagrama da estrutura computacional de uma *Idea* - extraído de Camargo *et al.* (2022).

bém, categorias auxiliares para facilitar a composição de uma *Idea* complexa. A categoria ***"QualityDimension"*** indica uma característica básica de uma propriedade, por exemplo, a propriedade posição pode ter as características X e Y. A categoria ***"TimeStep"*** indica um instante de tempo específico, atuando como um recorte temporal de uma região espacial. Uma *Idea* do tipo ***"TimeStep"*** é composta de ***"AbstractObjects"*** e ***"Property"*** que definem o estado dos objetos em um dado instante de tempo e uma sequência de ***"TimeSteps"*** pode ser utilizada para compor um ***"Episode"***.

O escopo de uma *Idea* permite distinguir se uma propriedade, objeto ou episódio representado por uma *Idea* é um elemento que realmente pertence à existência, se essa *Idea* diz respeito a um elemento hipotético que potencialmente poderia pertencer à existência, ou se é uma categoria/lei geral. Assim, o escopo de uma *Idea* pode ser: ***"Hypothesis"***, ***"Existence"***, ou ***"Category"***. Uma *Idea* com escopo ***"Existence"*** diz respeito a propriedades realmente adquiridas do ambiente, a objetos formados de fato por conjuntos dessas propriedades e episódios formados por conjuntos desses objetos que realmente ocorreram na existência, ou pelo menos que o agente que utiliza essas *Ideas* acredita que assim o seja. *Ideas* com o escopo ***"Hypothesis"*** referem-se a elementos que não ocorreram de fato na existência. Correspondem a elementos imaginários ou hipotéticos, sendo geradas tipicamente por mecanismos de planejamento, simulação mental, contrafactuais, viagem no tempo mental, entre outros. *Ideas* com o escopo ***"Categories"*** referem-se às *Ideas* que carregam informações de regularidades do ambiente e permitem a detecção de tipos de propriedades, objetos e episódios. As ***"Categories"*** são responsáveis por, ao longo do tempo, agregar regularidades do ambiente que indiquem a presença de fragmentos significativos da existência. Uma *Idea* do escopo ***"Category"*** deve implementar duas fun-

ções auxiliares. A função `membership(Idea)` retorna um valor entre 0 e 1 indicando o grau de pertencimento da *Idea* testada à categoria. A função `getInstance(Idea[])` gera uma instância (exemplo) dessa categoria. Opcionalmente, pode-se fornecer restrições para geração dessa instância através de uma lista de outras *Ideas*.

A Figura 3.6 apresenta os ícones utilizados para representar diferentes *Ideas*. Esses ícones são apenas uma ferramenta de visualização das estruturas de dados compostas por *Ideas*, onde cada ícone representa um tipo de *Idea* (***"Property"***, ***"Object"***, ***"Episode"***, ***"TimeStep"***, ***"QualityDimension"***) e diferentes cores representam os escopos (verde para ***"Existence"***, azul para ***"Hyposthesis"*** e vermelho para ***"Category"***).

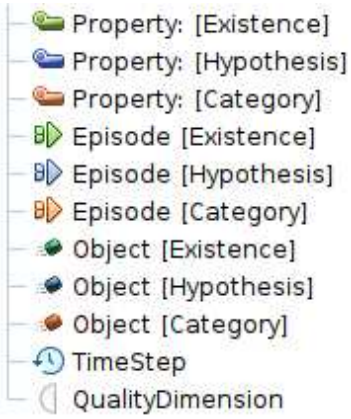


Figura 3.6 – Categorias e escopos de *Ideas* - adaptado de Camargo *et al.* (2022).

4 Sistema de Memória Episódica

Neste trabalho, o modelo teórico da arquitetura cognitiva proposta é inspirado em estudos de memória episódica humana. A Figura 4.1 apresenta a estrutura conceitual dessa arquitetura, onde os elementos retangulares representam *codelets* de processamento de informação e os círculos são *memory objects*, conforme a terminologia do CST. A arquitetura define mecanismos bases e a estrutura de módulos necessários para o reconhecimento de eventos, a combinação dos mesmos em episódios, a incorporação de episódios em uma memória de longo prazo e a recuperação de episódios dessa memória.

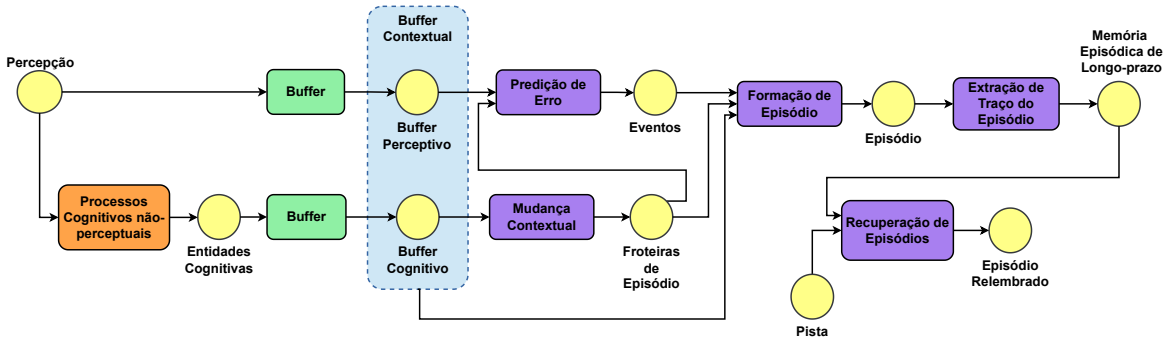


Figura 4.1 – Modelo conceitual para sistema de memória episódica proposto. Retângulos com cantos arredondados são *codelets* com setas indicando a direção do fluxo de informações; Círculos são *memory objects*; Retângulo azul com bordas tracejadas são grupos de memórias utilizado como uma ferramenta de organização para agregar *memory objects*.

Nas seções seguintes são apresentadas as definições de representação de conhecimento e mecanismos de operação dos principais componentes. O modelo teórico descreve um princípio de desenvolvimento de arquiteturas CST contendo sistema episódico. Aplicações específicas do modelo podem definir quais os algoritmos e mecanismos extras estarão presentes na arquitetura. O objetivo do modelo proposto é definir os elementos de representação e processamento necessários para a implementação de uma memória episódica.

4.1 Definições de Representação

O modelo proposto adota uma definição e representação específica para **eventos** e **episódios**. Tal abordagem é adotada para diferenciar informações de mudanças perceptuais simples e unimodais de informações de mudanças mais complexas, nas quais

múltiplas mudanças são agregadas em uma representação de mais alto nível e possivelmente (a depender da aplicação desenvolvida) multimodal. Inspirada na hierarquia de segmentações da *Teoria de Segmentação de Eventos* ilustrada na Figura 2.2, a divisão em dois níveis de segmentação temporal (eventos e episódios) auxilia também a incorporação da teoria de *Estabilidade Contextual* que será detalhada nas seções seguintes. A operação dos componentes descritos no modelo teórico deste capítulo é centrada na criação e manipulação dessas estruturas, permitindo sua codificação, armazenamento e recuperação desses elementos.

Eventos são unidades prototípicas de segmentação temporal. Cada evento representa um período de mudança nos valores das dimensões de propriedades de um objeto. Mais especificamente, um evento é uma *Idea* do tipo ***"Episode"*** (conforme nomenclatura de Camargo *et al.* (2022)) contendo índices para duas *Ideas* do tipo ***"TimeStep"***: uma para o início do evento, com uma anotação temporal e o valor da propriedade do objeto neste instante, e a segunda para o fim do evento, novamente com a anotação temporal e o valor da propriedade do objeto neste instante.

Adicionalmente, a *Idea* de um evento possui como valor uma *Idea* de **categoria de evento**, a qual indica informações sobre a classificação da transição ocorrida. Essa categoria de evento é responsável por descrever qual a dinâmica da transição dos valores do estado inicial ao estado final, por exemplo, diferenciando se a mudança foi linear ou quadrática, ou se a variação é acelerada ou constante? Dessa forma, um evento agrega uma interpretação sobre a transição ocorrida nas propriedades de um objeto. A seção 5.7 apresenta maiores detalhes sobre a operação das **categorias de eventos** adotadas na implementação de teste do modelo.

Episódios são representações estruturadas de um conjunto de eventos que compartilham um mesmo contexto. Os episódios integram diversos eventos e definem relações temporais entre eles, além de incorporar informações sobre a localização do agente quando o evento ocorreu e informações contextuais internas do agente provenientes dos demais processos cognitivos, como por exemplo mecanismos de definição de objetivos, emoções, planejamento, entre outros. Com isso, um episódio é a representação da cena experienciada pelo agente, contendo diversos eventos que compartilham um contexto espacial e/ou cognitivo semelhante. Em específico, adota-se uma representação em grafo para os episódio. Os episódios são representados por *Ideas* conectadas em grafo e são

descritos na seção 5.7.

Dado os modelos de representação de eventos e episódios adotados, as seções a seguir destacam os processos que cada componente do sistema episódico proposto realiza, realizando referências destacadas em negrito aos diferentes componentes do modelo na Figura 4.1.

4.2 Percebendo o Ambiente

O modelo de sistema episódico apresentado na Figura 4.1 omite os componentes sensoriais e perceptuais que estariam presentes em uma implementação da arquitetura. Logo, é assumido que o *memory object* **percepção** fornece um fluxo contínuo de dados sensoriais interpretados, ou seja, *Ideas* de objetos e suas propriedades identificados no ambiente ou propriocepção. Sempre que a memória perceptiva é lida por algum *codelet* da arquitetura, ela fornecerá um *snapshot* dos objetos percebidos pelo agente naquele instante. Os objetos detectados recebem também uma identificação única, a qual não se altera durante diferentes leituras da memória, mesmo quando saem e retornam ao campo de percepção do agente. Isso poderia ser realizado por métodos de segmentação e detecção de objetos e adicionalmente o agente pode possuir mecanismos de aprendizagem para descoberta de novas categorias de objetos e segmentação/clusterização de categorias existentes.

Os **processos cognitivos não-perceptuais** são as demais funções realizadas pelo agente, como planejamento, estado afetivo, raciocínio, entre outros. Tais componentes geram diversas informações que podem estar distribuídas pela implementação de uma arquitetura, porém são resumidas aqui como um único *memory object* denominado **entidades cognitivas**.

Os *codelets* de **buffer** operam acumulando a informação presente nas memórias de percepção e entidades cognitivas em um vetor, onde cada posição é uma cópia do estado destas memórias. Assim, tem-se à disposição um histórico recente da experiência do agente. Ambos os componentes de buffer possuem o mesmo comportamento, porém é importante selecionar as entradas que estarão conectadas ao *codelet* de buffer para a memória **entidades cognitivas**. Processos cognitivos podem ter memórias intermediárias, as quais não necessariamente são relevantes para o processamento do episódio, criando um uso desnecessário de memória ou informação duplicada.

4.3 Segmentação de Eventos e Episódios

O *codelet* **predição de erro** utiliza a memória do **buffer perceptivo** para realizar a segmentação de eventos observados. Este processo é inspirado na teoria de segmentação de eventos e utiliza *Ideas* como modelos de eventos para detecção de erros de predição nas mudanças que ocorrem nos valores de propriedades de objetos. Para cada objeto único, é mantida uma sequência dos últimos n valores observados de suas propriedades, e então *Ideas* de categorias de eventos são utilizadas para determinar se houve o início ou término de um evento. Uma categoria de evento deve definir um conjunto de propriedades de objetos e como a variação dos valores de suas dimensões deve ocorrer (linear, exponencial, discreta, etc.) ao longo do tempo. Ao receber, então, os últimos n valores observados de um objeto, a categoria irá selecionar as propriedades relevantes e verificar se as mesmas seguem a variação temporal definida. Dessa forma, o *codelet* de predição de erro utiliza as categorias de eventos presentes na memória do agente para monitorar as transições de início e fim de eventos (i.e. quando a sequência $obj_{t-n}, ..., obj_t$ não é detectada como pertencente a categoria de evento, porém a sequência $x_{t-n+1}, ..., x_{t+1}$ é, ou vice-versa). O período fechado que contém o início e fim de uma variação específica na propriedade de um objeto caracteriza um evento. Assim, é construída uma representação do evento formada por uma *Idea* do tipo ***"Episode"*** contendo dois ***"TimeStep"***, o primeiro contendo o objeto com valor da propriedade no início do evento e o segundo contendo o mesmo objeto com o valor da propriedade no final do evento, conforme a descrição da seção 3.2.

Paralelamente à detecção de eventos, o *codelet* **mudança contextual** utiliza a memória do **buffer cognitivo** para detectar segmentações de episódios. Este processo é inspirado pela teoria de estabilidade contextual e incorpora a descoberta de células de fronteira e evento de Zheng *et al.* (2022), as quais são ativadas na detecção de transições fracas e fortes, respectivamente. Assim, as segmentações por mudança contextual são dadas por fronteiras de episódios, as quais são mudanças significativas no padrão do contexto interno do agente. Ademais, as fronteiras de episódios podem ser do tipo fraca ou forte.

A fronteira fraca é caracterizada por uma mudança pequena de contexto, como por exemplo, a troca de uma sala para outra , transição entre passos de um plano em exe-

cução, surgimento ou desaparecimento de elementos do ambiente, entre outras. A definição dos fatores relevantes para detecção de fronteiras fracas é uma característica específica da aplicação, logo pode variar de uma implementação do modelo teórico proposto para outra. A detecção de uma fronteira fraca causa a segmentação forçada dos eventos atuais sendo rastreados, ou seja, todos os eventos atuais sendo detectados pelo *codelet* de **predição de erro** são finalizados e novos eventos são iniciados. Isso garante que os eventos possuem o correto contexto na sua representação dos episódios. Por exemplo, um agente se movendo em linha reta, resulta em somente um evento em que a propriedade posição é alterada linearmente, porém durante esse movimento o agente pode transitar de uma região relevante a outra. Com isso, pode ser interessante que a representação do segmento do evento executado na primeira região seja diferente da representação na segunda região. Porém, a categoria de evento de movimento não leva em consideração a região do agente e dessa forma a segmentação deve ser forçada pela fronteira fraca.

A fronteira forte é uma mudança significativa de contexto que causa a segmentação do episódio, como por exemplo, uma troca de objetivo, troca de planejamento, um estímulo afetivo forte e repentino, entre outros. A fronteira forte é responsável por sinalizar a finalização da construção do episódio atual e o início da construção de um novo episódio. Portanto, pode ser relevante que um dos fatores de detecção de fronteira forte seja o tempo decorrido desde a última fronteira, garantindo que os episódios não sejam muito extensos.

O *codelet* de **formação do episódio** agrega uma lista de eventos detectados com o contexto cognitivo interno do agente em uma única representação. Essa representação apresenta a experiência do agente em um certo período de tempo. Como descrito anteriormente, o início de um episódio se dá com uma fronteira forte. Após isso, os eventos que são inseridos no *memory object* **eventos** pelo *codelet* de predição de erros são incorporados ao episódio atual. Nesse processo são criadas conexões entre a *Idea* do novo evento e as *Ideas* dos eventos já presentes no episódio, de forma a indicar as relações temporais existente entre elas. Para isso, é utilizada a lógica intervalar de Allen (ALLEN, 1983). A Figura 4.2 apresenta as sete relações consideradas entre dois intervalos de tempo. A álgebra intervalar de Allen considera 13 relações no total, porém 6 delas são inversões das demais apresentadas (com exceção de ‘X igual Y’) e por isso podem ser diretamente inferidas a partir da existência da sua relação oposta. Por exemplo, a relação ‘Y é prece-

dido por X' é a inversa de 'X precede Y', ambas indicam a mesma sequência temporal de intervalos, porém uma indica a relação de Y para X e a outra de X para Y.

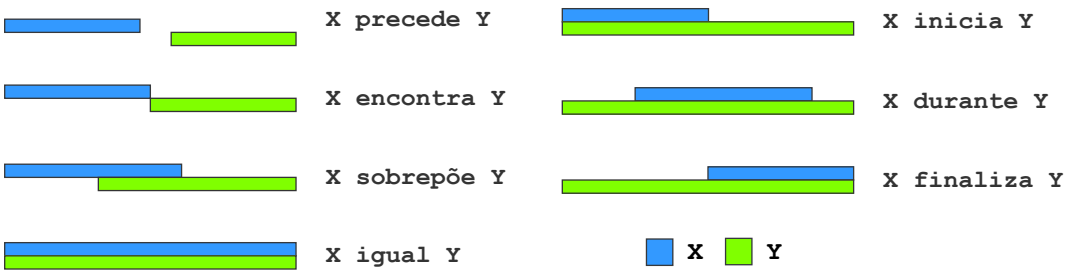


Figura 4.2 – Possíveis relações temporais entre dois intervalos de tempo.

Após a inserção das relações temporais, os eventos também são conectados com seu contexto. O contexto é um estado mental do agente, incluindo as entidades percebidas, sua motivação, planejamento, entre outros processos cognitivos. O papel do episódio é associar o contexto atual com os eventos segmentados em uma unidade de representação. Tal informação é adquirida do **buffer contextual** que engloba os *memory objects* **buffer perceptual** e **buffer cognitivo**, a qual permite a identificação dos elementos do contexto relevantes no período em que o evento ocorreu.

Informações de contexto são associadas por links a cada evento do episódio. O conteúdo dessas informações varia a depender da aplicação, pois o contexto deve ser algo relevante para o agente realizar sua interpretação de uma experiência. Assim, as escolhas de quais informações são utilizadas para detecção de fronteiras fortes e fronteiras fracas e a composição do contexto dos eventos estão relacionadas entre si. Logo, assumimos uma representação básica de um episódio como grafo heterogêneo conforme a Figura 4.3.

Vamos tomar como exemplo um agente que se locomove entre regiões de um ambiente e deve cumprir algumas tarefas. Para isso, este agente é capaz de detectar a região atual, representar e escolher um objetivo para cumprir sua lista pré-determinada de tarefas e sensorear informações proprioceptivas e do ambiente. Assim, podemos escolher a transição de uma região para outra como sendo uma fronteira fraca, pois, embora a informação de qual é a região do agente ajude a interpretar os eventos observados, essa transição não é significativa o suficiente para indicar uma segmentação de episódios. Enquanto que, a transição de um objetivo para outro é escolhida como sendo uma fronteira forte, pois durante o período em que o objetivo do agente permanece fixo, os eventos que ocorrem no ambiente (os quais incluem as próprias ações do agente, já que sua propri-

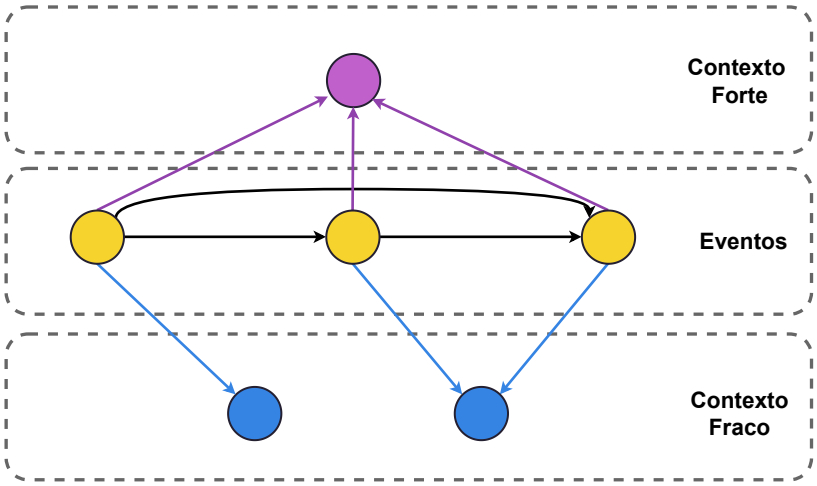


Figura 4.3 – Exemplo de representação de um episódio. Cada círculo e hexágono corresponde a uma *Idea* que contém informação sobre um objeto, evento, região, motivação, ou posição.

ocepção pode estar presente como um objeto no *memory object* **percepção**) dependem causalmente das ações, e por consequência objetivo, do agente. Dessa forma, ao construir o episódio é importante que a informação da região e do objetivo façam parte do contexto. Isso irá permitir que o agente interprete os eventos do episódio quando o mesmo for relembrado, pois somente a sequência das mudanças nos objetos não é importante para o agente caso o mesmo não saiba onde e por que ocorreram.

Em comparação com a Figura 4.3, a região onde ocorreu um evento é um contexto associado as fronteiras fracas, enquanto o objetivo ativo durante o evento é um contexto associado as fronteiras fortes. Logo, o contexto fraco pode se alterar de um evento ao outro, pois fronteiras fracas não forçam a segmentação de episódios, somente de eventos. Observa-se também com isso que um evento não irá possuir duas informações distintas de um mesmo tipo de contexto fraco, no exemplo dado, um evento nunca irá ocorrer em duas regiões distintas, pois a troca de região é uma fronteira fraca que força a segmentação dos eventos. Já o contexto forte não se altera entre os eventos de um episódio, pois isso iria causar a segmentação do episódio em si.

Temos então, uma representação de episódio que carrega informações de o que ocorreu (*what*) utilizando os eventos, quando ocorreu (*when*) utilizando relações temporais entre eventos, onde ocorreu (*where*) utilizando o contexto fraco contendo informação da região do agente e do porquê ocorreu (ainda que parcialmente) utilizando o contexto forte contendo informação do objetivo ativo do agente. A representação de episódio permite,

dessa forma, representar a informação WWW de Tulving (1972), além de possibilitar a incorporação de outras informações de contexto, como o 'porquê' no exemplo anterior.

4.4 Incorporação e Recuperação de Memórias Episódicas

Quando uma fronteira forte de episódio é detectada, acarretando no término do episódio atual e início da codificação de um novo episódio, o episódio concluído é transferido para a memória de longo prazo. Para isso, a representação do mesmo é adaptada e integrada com os episódios já armazenados. Esse processo é executado pelo *codelet* de **extração de traço do episódio**, onde a informação do episódio é abstraída e conectada às demais informações presentes na memória de longo-prazo. Alternativamente, a acomodação dos novos episódios pode ser realizada durante períodos de pouca atividade do agente.

A abstração do episódio consiste na conversão das instâncias de objetos e eventos em categorias. A escolha da representação de categorias é dependente da implementação. Pode-se também, adotar um processo de filtragem dos elementos do episódio a serem armazenados, utilizando-se, por exemplo, mecanismos atencionais que atribuam relevância aos objetos e/ou eventos detectados. A seção 5.7 apresenta o processo adotado para a implementação teste.

O *codelet* de **recuperação de episódios** permite, a partir de uma pista contendo partes da informação de um episódio, a busca por episódios semelhantes na memória episódica de longo prazo. O resultado da busca pode retornar um ou mais episódios, os quais são inseridos no *memory object* **episódio relembrado**. Em caso de falha, nenhum episódio é retornado. O episódio também deve ser reconvertido para a representação utilizada pelo *codelet* **formação de episódio**, ou seja, ter a estrutura de representação igual a de um episódio experienciado, desfazendo assim, a adaptação feita pelo *codelet* **extração do traço de episódio**.

5 Materiais e Implementação

O Capítulo 4, apresentado anteriormente, introduz as considerações teóricas do sistema de memória episódica proposto. Esse modelo é uma base para a implementação de arquiteturas cognitivas com sistema de memória episódica, indicando quais os processos que devem ser executados pela arquitetura, como eles se conectam e qual a representação da informação que eles trocam. Neste capítulo, é apresentado uma implementação utilizando esse modelo.

Inicialmente, é introduzido o ambiente de simulação para o qual deseja-se desenvolver um agente. Adicionalmente, a biblioteca WS3D-Coppelia foi desenvolvida durante a execução desse trabalho para operar como controlador da simulação e interface com a arquitetura que controla o agente.

Então, é apresentada a arquitetura implementada, onde são detalhados os seus componentes e os algoritmos utilizados para executarem os processos descritos pelo modelo base. A implementação é feita com a ferramenta CST e representação *Ideas*.

5.1 Simulador CoppeliaSim

O software CoppeliaSim[1] (ROHMER *et al.*, 2013) é um simulador 3D voltado para o desenvolvimento e teste de projetos robóticos. O CoppeliaSim oferece diferentes motores de física[2], interfaces para controle da simulação por código, componentes para simulação de sensores e atuadores, *plugins*, entre outras funcionalidades. O simulador é oferecido com uma licença gratuita para entidades educacionais e uso não-comercial.

5.1.1 Elementos de Simulação

Uma simulação no CoppeliaSim é composta por uma **cena** e múltiplos **modelos**. Um **modelo** é um conjunto de objetos organizados em uma árvore hierárquica,

[1] Nome atual do software anteriormente chamado de V-REP: <https://www.coppeliarobotics.com>

[2] Tradução para o termo técnico em inglês *physics engine*, que designa o componente de software responsável por simular as leis da física em um ambiente virtual. Existem diferentes motores de física disponíveis, que podem variar quanto ao realismo da simulação (atrito, distorções elásticas em colisões, diferentes tipos de materiais, etc) e em seu desempenho computacional. O nível de realismo, entretanto, tem seu preço, e motores de física mais realistas serão mais lentos e ocuparão mais tempo de simulação. Existem motores de física *open-source* e outros desenvolvidos por empresas especializadas, que podem demandar uma licença paga.

constituindo um sub-elemento de uma **cena**. Os objetos que compõem um **modelo** podem ser de diversos tipos, como formas geométricas, juntas, sensores de visão, proximidade ou torque, luzes, entre outros. Dessa forma, o **modelo** de um braço robótico poderia ser composto por objetos de formas geométricas para formar seus braços e juntas como atuadores dos eixos.

Uma **cena** é constituída por um ambiente, um *script* principal e visualizações, além de objetos e outros modelos. Um ambiente determina parâmetros de renderização da cena, como cor de fundo e luminosidade ambiente. O *script* principal é responsável por inicializar a simulação de todos os componentes da cena. Visualizações são janelas para exibição do conteúdo de objetos com visualizações associadas, como câmeras e sensores de profundidade.

5.1.2 *Scripts*

A simulação do CoppeliaSim é customizável e permite o uso de *scripts* para programação de comportamentos específicos. Os *scripts* são códigos que podem ser associados a elementos específicos da simulação e podem ser definidos em linguagem Lua ou Python, sendo executados por meio de funções de *callback* chamadas pelo simulador. As funções de callback possuem nomes fixos e estão disponíveis aos *scripts* dependendo do elemento ao qual estão integrado (e.g., a função `sysCall_init()` é chamada durante a inicialização da simulação). Um *script* pode ser integrado a uma cena ou modelo para a execução de um código específico para o comportamento do mesmo. Esse *script* é salvo e carregado junto do modelo. O *script* integrado deve ser responsável pelo comportamento específico do modelo associado a ele, por exemplo, o modelo de um robô pode ter um *script* integrado que lê os valores de seus sensores, toma uma decisão e executa ações nos seus atuadores.

5.1.3 API Remota

Um dos métodos para controle da simulação é a API remota do CoppeliaSim. A API permite a comunicação de aplicações externas e é compatível com as linguagens Python, C++, Matlab, Octave, Java, Lua e Rust, utilizando sempre as mesmas chamadas de funções. As funções da API remota permitem o controle completo de qualquer elemento (cena e modelos) e da simulação e seus parâmetros, além da leitura e escrita de arquivos e uso de plugins.

5.2 Biblioteca de Simulação

O ambiente de simulação foi implementado no CoppeliaSim, através da definição de uma cena e modelos de elementos específicos. Adicionalmente, a API remota do CoppeliaSim é utilizada para criação de uma biblioteca de configuração e controle de simulação denominado **WS3D-Coppelia**[3]. O WS3D-Coppelia é uma refatoração de um servidor de mundos virtuais, chamado de WS3D (WorldServer 3D), utilizado por nosso grupo de pesquisa em diversos trabalhos anteriores do grupo. O WS3D-Coppelia foi desenvolvido como parte desse trabalho de pesquisa especificamente para o desenvolvimento dos experimentos desse trabalho, e deverá substituir o WS3D nos próximos trabalhos do grupo de pesquisa.

5.2.1 Elementos Simulados

O principal elemento a ser simulado no ambiente é um agente. O modelo do agente, apresentado na Figura 5.1, é composto por uma forma geométrica (em amarelo) customizada para representar seu corpo, um sensor visual (em azul) para sensoriamento do ambiente e uma base circular para simulação de colisões com outros objetos do ambiente (cinza transparente). Os elementos do sensor visual e base circular não são exibidos durante a simulação, mas continuam interagindo com o ambiente.

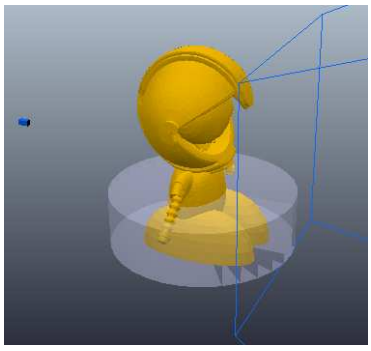


Figura 5.1 – Modelo de renderização do agente simulado.

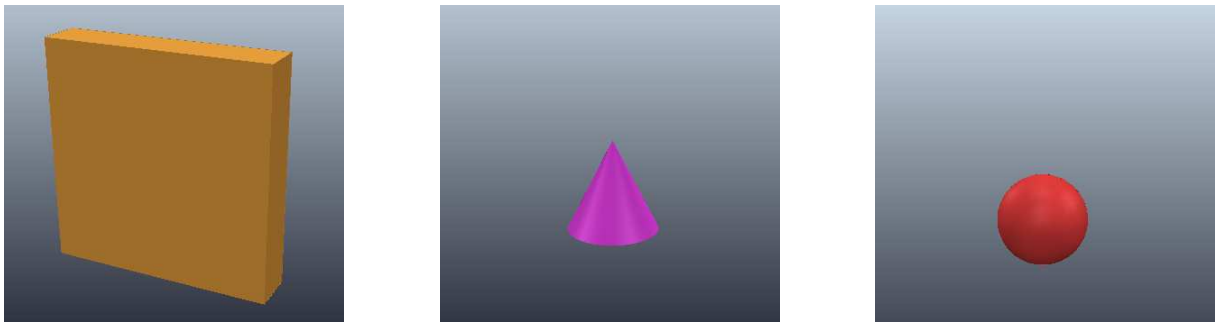
Três tipos de objetos (Figura 5.2) podem estar presentes no ambiente de simulação:

(a) **Paredes**: são blocos retangulares utilizados para criar obstáculos no ambiente. Podem ser criados nas cores branco, azul, vermelho, verde, magenta, amarelo, laranja, cinza ou marrom.

[3] Seu código encontra-se disponível em: <https://github.com/CST-Group/WS3D-Coppelia>.

(b) **Joias**: são cones representando itens de interesse que os agentes podem coletar no ambiente. Podem ser criados nas cores branco, azul, vermelho, verde, magenta ou amarelo.

(c) **Alimentos**: são esferas que fornecem energia ao agente quando consumidas.



(a) Parede (b) Joia (c) Alimento

Figura 5.2 – Exemplo de modelos dos elementos simulados.

Neste trabalho é considerado para todos os elementos de simulados, assim como o ambiente em si, que os valores de posição e tamanho estão em metro, as angulações em radianos e as componentes vermelho, azul e verde de propriedades de cores são dadas por porcentagens no intervalo $[0, 100]$.

5.2.2 O Agente

O modelo do agente possui um *script* integrado para sensoriamento e controle de sua posição no ambiente. Funções auxiliares permitem definir os parâmetros para execução das ações de mover, rotacionar, pegar objeto e comer alimento. Essas funções não são chamadas pelos callbacks do CoppeliaSim, porém podem ser executadas através da API remota. Um agente possui três informações principais que podem ser monitoradas através da biblioteca desenvolvida: posição, orientação e energia. A posição é a coordenada (x, y) do agente no ambiente, a orientação é a angulação do eixo do agente, que aponta na direção de seu campo de visão, em relação ao eixo X de referência da simulação, e a energia é um valor entre 0 e 1000 que decai linearmente durante a execução da simulação e ao atingir 0 impede que o agente execute mais ações.

A biblioteca WS3D-Coppelia oferece a classe `Agent` para o controle de um agente simulado. Destacam-se as seguintes funções:

- **`getPosition()`** retorna a posição (X, Y) do agente no ambiente;

- **`getPitch()`** retorna o ângulo do agente em relação ao eixo X de referência do ambiente em radianos;
- **`getFuel()`** retorna o nível de energia atual do agente;
- **`moveTo(x,y)`** envia os comandos para o simulador de forma a mover o agente para a posição definida;
- **`eatIt(Thing)`** consome o objeto recebido se o mesmo for um alimento;
- **`rotate()`** rotaciona o agente no sentido anti-horário até receber outro comando;
- **`sackIt(Thing)`** adiciona o objeto do ambiente ao inventário interno do agente.

Agentes são inicializados com um conjunto de objetivos de coleta de joias. Esses objetivos são denominados *leaflets* e funcionam como tabelas de troca de joias por pontos. Um *leaflet* determina quantas joias de cada tipo devem ser coletadas e quantos pontos serão fornecidos durante a troca. Os *leaflets* são únicos para cada agente e cada agente possui três diferentes *leaflets*. A lista de *leaflets* de um agente pode ser acessada pela função `getLeaflets()`, ou renovada (gerado um novo conjunto de *leaflets*) a partir da função `generateNewLeaflets()`.

Para completar um *leaflet* (e obter o correspondente número de pontos do *leaflet*) o agente deve coletar as joias necessárias do ambiente com a ação `sackit(Thing)` e então, quando completo, executar a ação `deliverLeaflet(leafletId)`. Assim, as joias coletadas pelo agente serão consumidas e o agente recebe a pontuação referente ao *leaflet*.

Os objetos da classe `Leaflet`, contêm as informações referentes aos requisitos para completar um *leaflet*, assim como seu progresso atual.

5.2.3 Objetos

Os objetos estáticos do ambiente são representados pela classe `Thing` da biblioteca **WS3D-Coppelia**. A classe `Thing` possui um atributo que determina seu tipo entre joia, parede e alimento. Os métodos `isJewel`, `isBrick` e `isFood` podem ser invocados para verificar o tipo de um `Thing`. Adicionalmente, destaca-se alguns métodos auxiliares para acesso a atributos do elemento simulado:

- `getPos()` retorna a posição (X, Y) do objeto;

- `getColor()` retorna a cor RGB do objeto;
- `isInOccupancyArea(x,y)` determina se uma posição (X, Y) está na região ocupada pelo objeto, sendo útil para determinar se uma joia ou alimento está ao alcance do agente.

5.2.4 Controlador WS3DCoppelia

O acesso principal para controle da simulação é feito pela classe `WS3D-Coppelia`. Ao ser instanciada, é estabelecida uma conexão com o simulador CoppeliaSim, permitindo o controle da simulação e a criação de agentes e objetos.

As funções de controle de simulação são:

- `startSimulation()` inicializa a simulação, adicionando todos os agentes e objetos criados;
- `stopSimulation()` finaliza a simulação;
- `createAgent(x,y,[color])` cria um novo agente na posição (X, Y) e retorna uma instância da classe Agent. Opcionalmente uma cor entre amarelo, verde, magenta e vermelho pode ser definida;
- `createThing(category,x,y)` cria um objeto na posição (X, Y) definida e retorna uma instância da classe `Thing` para controle do mesmo. A categoria do objeto define se o mesmo será uma joia, alimento ou parede. Adicionalmente, existe uma categoria para cada cor de objeto;
- `createBrick(type,x1,y1,x2,y2)` cria uma parede do tipo especificado com diagonal definida pelos pontos $(X1, Y1)$ e $(X2, Y2)$ e retorna uma instância da classe `Thing`. O tipo da parede define sua cor, podendo ser vermelha, azul, verde, amarela, magenta, branca, laranja, cinza ou marrom.

5.3 Descrição da Arquitetura

A Figura 5.3 apresenta o diagrama estrutural de todos os componentes da implementação. A arquitetura pode ser dividida em três grandes blocos, destacados na Figura 5.3 por regiões sombreadas. Cada bloco corresponde a instanciação de uma parte do modelo base da Figura 4.1. Destacado em laranja, na parte superior esquerda, o sistema

de percepção da arquitetura é responsável por implementar os mecanismos de detecção de objeto de forma a fornecer as informações perceptuais que são utilizadas pelo sistema episódico, correspondendo ao *memory object* percepção do modelo base. Em verde, na parte superior direita, diferentes processos cognitivos são utilizados para processamento da informação perceptual, planejamento e execução de ações pelo agente. Esses processos correspondem ao *codelet* de processos cognitivos não-perceptuais e ao *memory object* entidades cognitivas do modelo base. Destacado em azul, na parte inferior do diagrama, o sistema episódico consiste nos demais processos do modelo base que são responsáveis pela segmentação e processamento de eventos e episódios.

Os *codelets* da arquitetura são codificados por cor para indicar grupos de *codelets* que possuem comportamento semelhante. A arquitetura possui os seguintes tipos de *codelets*:

- ***Codelets* sensoriais (verde)** adquirem informações do ambiente através dos sensores disponíveis ao agente. Estes *codelets* possuem acesso ao ambiente, podendo fazer requisições às funções do simulador. *Codelets* sensoriais somente transferem os dados adquiridos para as memórias de saída, não realizando conversão para uma representação na forma de *Ideas*;

- ***Codelets* perceptivos (vermelho)** processam a informação da memória sensorial para identificar propriedades e objetos do ambiente. As entidades detectadas são estruturadas como *Ideas* e inseridas na memória de saída;

- ***Codelets* de aprendizado perceptual (vermelho)** armazenam para cada objeto único, o último estado observado de suas propriedades. Adicionalmente, os objetos na memória perceptual de longo-prazo possuem um valor de novidade no intervalo [0,1], iniciando-se em 1 quando o objeto é percebido pela primeira vez (e armazenado na memória), e decaindo ao longo do tempo. Sempre que o objeto é percebido novamente, sua novidade é incrementada de forma proporcional à diferença entre o valor atual e o máximo 1;

- ***Codelets* de impulso (azul claro)** monitoram as memórias perceptuais para detectar a presença de objetos capazes de satisfazer alguma necessidade do agente. Os impulsos gerados possuem um valor de intensidade associado e permanecem armazenados na memória de impulsos até serem satisfeitos;

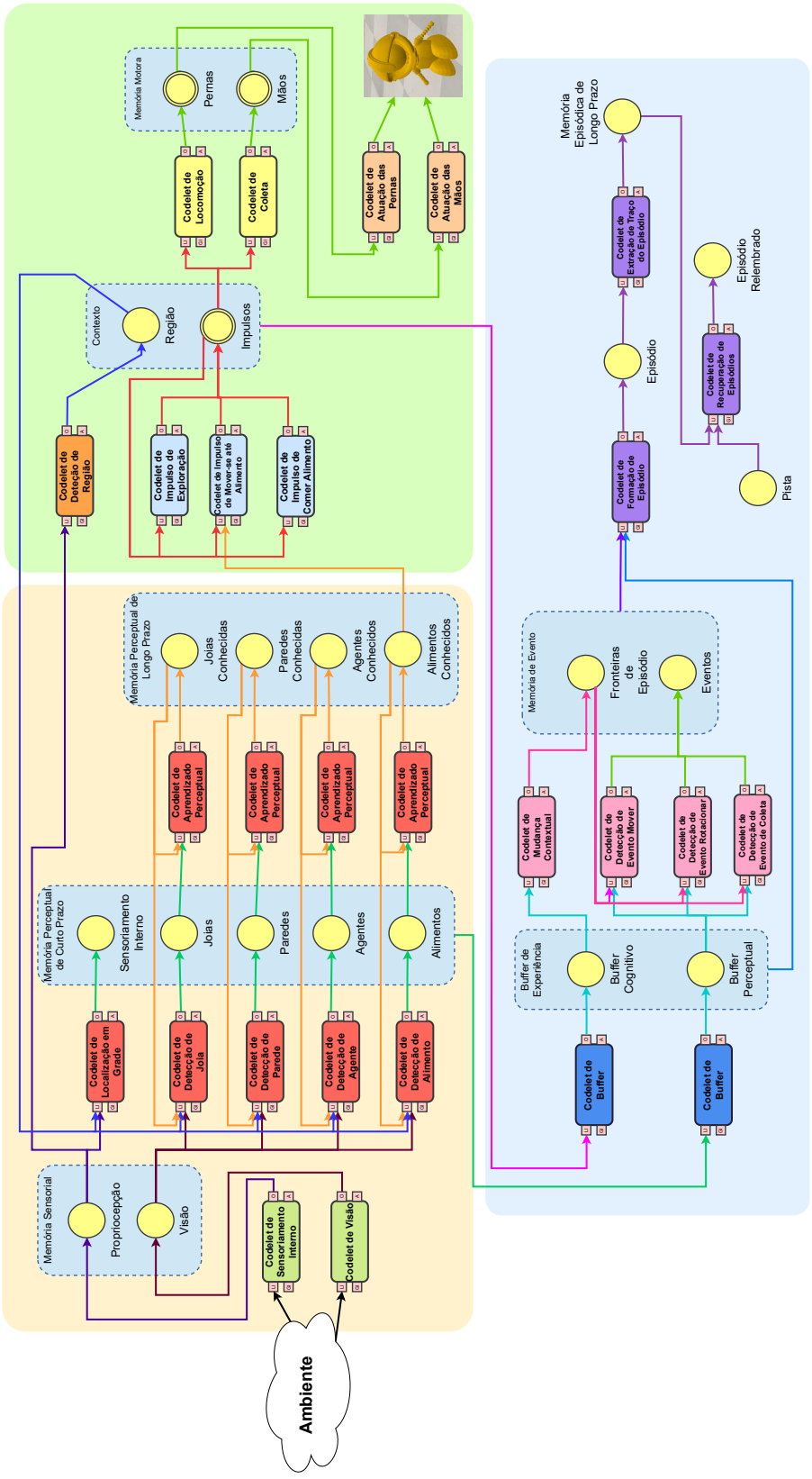


Figura 5.3 – Diagrama da arquitetura cognitiva implementada.

- ***Codelet* de detecção de região (laranja)** determina a partir da posição do agente e de uma lista de categorias de regiões qual a região atual do agente;

- ***Codelets* de comportamento (amarelo)** determinam as ações necessárias para satisfazer o impulso de maior intensidade. A ação atual é armazenada nas memórias motoras;
- ***Codelets* de atuação (laranja claro)** enviam os comandos apropriados ao agente para execução da ação atual;
- ***Codelets* de buffer (azul)** acumulam as últimas n informações das memórias de entrada em um vetor;
- ***Codelets* de detecção de eventos e fronteiras (rosa)** realizam a identificação de eventos e mudanças de episódios;
- ***Codelets* do sistema episódico (roxo)** são responsáveis pela formação, armazenamento e recuperação de episódios.

5.4 Componentes Perceptivos

O sistema perceptivo do agente engloba os *codelets* sensoriais, perceptivos e de aprendizagem perceptual. Os sensores do agente possuem acesso direto ao ambiente simulado. Dessa forma, os módulos perceptivos não são responsáveis por realizarem a detecção de objetos do ambiente, pois as informações dos objetos presentes no alcance do sensor visual do agente podem ser diretamente acessadas pelos *codelets* sensoriais. Dois *codelets* sensoriais estão presentes na arquitetura. O ***codelet* de propriocepção**[4] adquire as informações de posição, orientação e energia do agente controlado. O ***codelet* de visão** adquire uma lista com todos os objetos atuais na visão do agente. Essa lista possui instâncias da classe `Thing` e `Agent` da biblioteca de simulação descrita na seção 5.2, podendo ser um objeto ou agente.

Com as informações sensoriais obtidas, os *codelets* perceptivos estruturam *Ideas* para cada objeto do ambiente, conforme a categoria do mesmo. Os *codelets* perceptivos sobrescrevem as informações presentes na memória de saída. Assim, as *Ideas* presentes nas memórias são referentes a um *snapshot* das informações perceptuais atuais do agente e não são persistidas entre execuções dos *codelets* perceptivos. Os ***codelets* de detecção de joia, parede, agente e alimento** utilizam a lista de elementos detectados

[4] Nomes de *codelets* e *memory objects* em negrito referem-se a elementos da Figura 5.3.

pelo sensor de visão para identificar objetos pertencentes a cada categoria e inserir na memória de saída uma *Idea* contendo as informações dos mesmos. O ***codelet* de localização em grade** é responsável por criar uma *Idea* com as informações de propriocepção do agente.

Um elemento importante na representação de eventos e episódios adotada na implementação desse trabalho é a localização espacial de agentes e objetos. A arquitetura implementada utiliza uma representação discreta e hierárquica, constituída de uma grade hexagonal e um mapa de regiões conectadas. Assim, os *codelets* perceptivos realizam também a transformação da posição do agente controlado e de todos objetos percebidos para a representação em grade hexagonal adotada.

5.4.1 Representação Espacial Hierárquica

Para implementação da arquitetura, optou-se por uma representação espacial do ambiente conhecido pelo agente na forma de regiões retangulares. Assim, os episódios codificados pelo agente possuem um contexto fraco na forma de um contexto espacial contendo a informação da região em que o mesmo se encontra. Adicionalmente, é considerado que uma região é subdividida por uma grade hexagonal que permite a identificação das posições dos objetos dentro do ambiente de forma relativa através de coordenadas (u, v).

Assim, a posição dos objetos é relativa à região em que o agente se encontra, sendo determinada por uma grade hexagonal centrada na região. Cada célula da grade é identificada por uma coordenada matricial (u, j), conforme apresentado na Figura 5.4, onde as colunas de índice ímpar são deslocada para cima em relação a uma grade quadriculada equivalente. Dessa forma, a posição de um objeto é dada pela coordenada (u, j) da célula ocupada pelo mesmo na grade hexagonal. Além disso, se o objeto se estende por mais de uma célula, todas são incluídas na *Idea* que representa o objeto.

Para isso é assumido que o ambiente é dividido em regiões conectadas, sendo essas pré-determinadas durante a codificação da arquitetura. A grade hexagonal possui um tamanho fixo e menor do que todo o mapa da simulação, porém a grade recebe um *offset* de forma que a célula com coordenada $(0, 0)$ esteja sempre no centro da região em que o agente se encontra (vide Figura 5.5). Logo, a posição espacial de um objeto é representada relativamente à região ocupada pelo agente e para determinar a posição absoluta de um objeto detectado, deve-se conhecer a coordenada da célula e a região em que o agente se encontra.

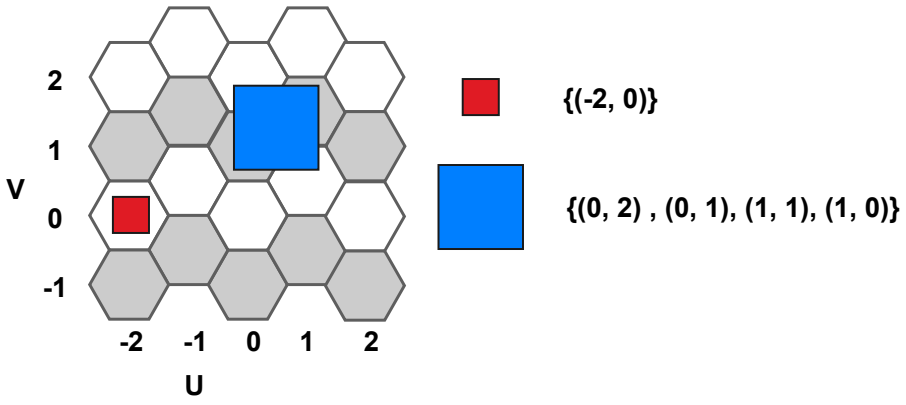


Figura 5.4 – Representação matricial da grade hexagonal. De forma alternada, é destacado em cinza as células pertencentes a uma mesma linha. Adicionalmente é exibido como pode-se obter o conjunto de células ocupadas por objetos do ambiente. O objeto vermelho ocupa somente a célula $(-2, 0)$, enquanto do objeto azul ocupa as células $(0, 2)$, $(0, 1)$, $(1, 1)$ e $(1, 0)$

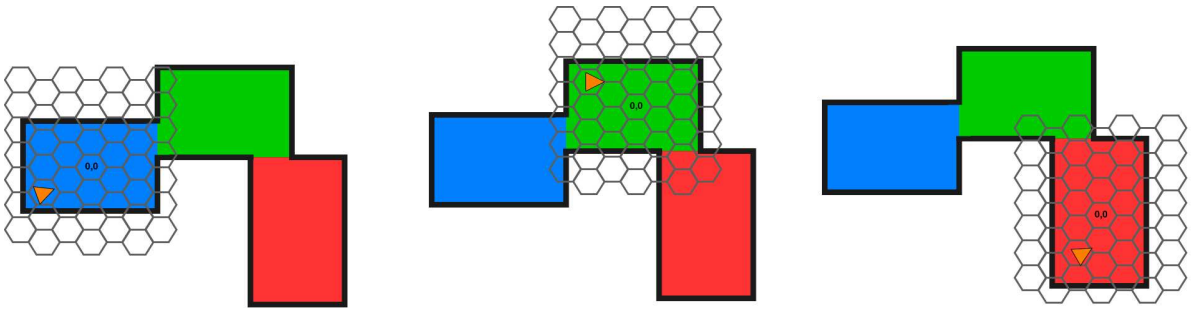


Figura 5.5 – Exemplo de *offset* da grade hexagonal por região. Cada retângulo representa uma região conhecida pelo agente (triângulo laranja). Quando posicionado no ambiente azul, a grade hexagonal é centrada no mesmo. Conforme o agente se locomove pelo ambiente a grade para representação da posição dos objetos percebidos é deslocada de forma a alinhar seu centro como o centro do ambiente.

5.4.2 Aprendizado Perceptual

Por fim, os ***codelets* de aprendizado perceptual** armazenam os últimos estados de cada objeto único detectado. O simulador atribui a cada objeto um ID, o qual é incorporado à *Idea* de cada objeto detectado pelos *codelets* de detecção de objetos. Com isso, é mantido uma cópia de longo-prazo das memórias perceptuais do último estado observado de cada objeto, onde, a cada nova detecção de um mesmo objeto, o seu estado anterior salvo em memória é sobrescrito pelo estado atual, permitindo, assim, que o agente tenha informações sobre objetos detectados no ambiente que não se encontram mais em seu campo de visão.

5.5 Módulos Motivacionais

O sistema motivacional do agente é responsável por determinar o comportamento do agente. Isso é alcançado por meio da geração de impulsos, que surgem associados a objetos detectados do ambiente. O impulso indica o desejo do agente em executar um determinado comportamento em relação ao objeto, (e.g., mover-se até ele). A arquitetura contém três possíveis impulsos, relacionados a diferentes necessidades do agente. Cada tipo de impulso é gerado por um *codelet* e armazenado no *memory container* **impulsos**. Como visto no Capítulo 3, no CST, um *memory container* é uma estrutura de memória em que diversos elementos podem ser inseridos, cada um deles com um diferente valor de avaliação (intensidade). Quando um *codelet* lê um *memory container*, ao invés de obter todos os elementos de memória inseridos no mesmo, ele obtém somente um deles, selecionado algum critério de relevância (maior, menor, aleatório, etc). Assim, as instâncias de impulsos gerados possuem uma intensidade associada, que é utilizada como valor de avaliação da informação quando armazenada no *memory container*. Quando uma leitura é realizada no *memory container*, retorna-se somente a informação de maior avaliação. Com isso, os demais *codelets* que acessam o *memory container* de impulsos recebem somente a *Idea* do impulso com maior intensidade.

O ***codelet* de impulso de exploração** refere-se ao desejo do agente em explorar as diversas regiões do ambiente. O impulso de exploração é associado ao objeto do próprio agente, gerado pela percepção proprioceptiva. Um impulso de exploração sempre irá existir na memória de **impulsos** do agente e, sempre que satisfeito, um novo impulso para exploração de um local aleatório do ambiente é criado.

O ***codelet* de impulso de mover-se até alimento** cria novas instâncias de impulsos quando o agente possui pouca energia. Nesse momento, impulsos relacionados aos alimentos armazenados na memória perceptual de longo-prazo são gerados. Cada objeto de alimento gera um impulso diferente de mesma intensidade. Dessa forma, a seleção do impulso ativo pelo *memory container* de impulsos é aleatória. Ou seja, quando está com pouca energia o agente possui o comportamento de selecionar um alimento aleatório dentre os que conhece para o qual se deslocar.

O ***codelet* de impulso de consumir alimento** surge quando há um alimento muito próximo ao agente. Ele ocorre mesmo quando o nível de energia do agente não está

baixo, de forma a impedir que o alimento bloqueie o movimento do agente.

5.6 Módulos de Atuação

O sistema de atuação do agente é responsável por executar os comportamentos criados pelo sistema motivacional, sendo subdividido em *codelets* de ação e *codelets* de atuação motora.

O ***codelet* de coleta** gera comandos para comer alimentos do ambiente. Quando o impulso de maior avaliação é do tipo 'consumir alimento', uma ação para comer o alimento alvo do impulso é enviada para o *memory container* **mãos**. O ***codelet* de locomoção** gera os comandos de movimentação do agente pelo ambiente. Quando o impulso ativo é relacionado ao desejo do agente em estar em uma posição específica do ambiente, o *codelet* de locomoção realiza o planejamento da rota até este local e envia a sequência de ações ao *memory container* **pernas**.

5.6.1 Planejamento de Rotas

O planejamento de rotas realizado pelo agente é hierárquico e baseia-se na forma de representação de localização espacial adotada. A construção de rotas hierárquicas é explorada em diferentes trabalhos da literatura (MAC *et al.*, 2017; ZUO *et al.*, 2015; GREGORIĆ *et al.*, 2023). Em especial, a implementação deste trabalho baseia-se na abordagem geral de Ryu (2020) (Figura 5.6), sem considerar as técnicas de construção de mapa propostas pelo mesmo. A rota é definida de modo global pela representação de alto nível das regiões do ambiente e suas adjacências, gerando uma sequência de sub-objetivos para o agente. Então, para a região atual do agente é planejada uma rota local para alcançar a próxima região da rota global.

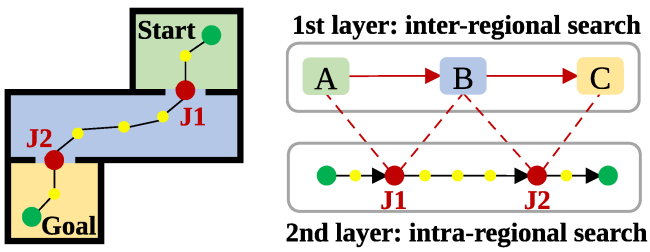


Figura 5.6 – Ilustração do método de planejamento de rotas hierárquico - extraído de Ryu (2020).

A *Idea* de cada região é uma categoria que define uma região retangular do

ambiente em coordenadas absolutas. Adicionalmente, a *Idea* contém índices para as demais *Ideas* de regiões adjacentes que podem ser alcançadas a partir dela. Com isso, o conjunto de *Ideas* de regiões conhecidas forma uma representação em grafo do ambiente. Cada região possui a informação de quais células da grade hexagonal dão acesso a uma região adjacente. Ou seja, a indicação de onde se encontram os pontos de junção entre as regiões de forma relativa à região atual.

As informações das regiões do ambiente permitem, então, definir as rotas global e local do agente, conforme ilustrado na Figura 5.7. A rota global é um caminho obtido do grafo de *Ideas* de regiões que parte da região atual do agente até a região referente a posição objetivo. A rota local é definida por uma sequência de células hexagonais que levam da posição atual do agente até a junção da região atual com a próxima região da rota global.

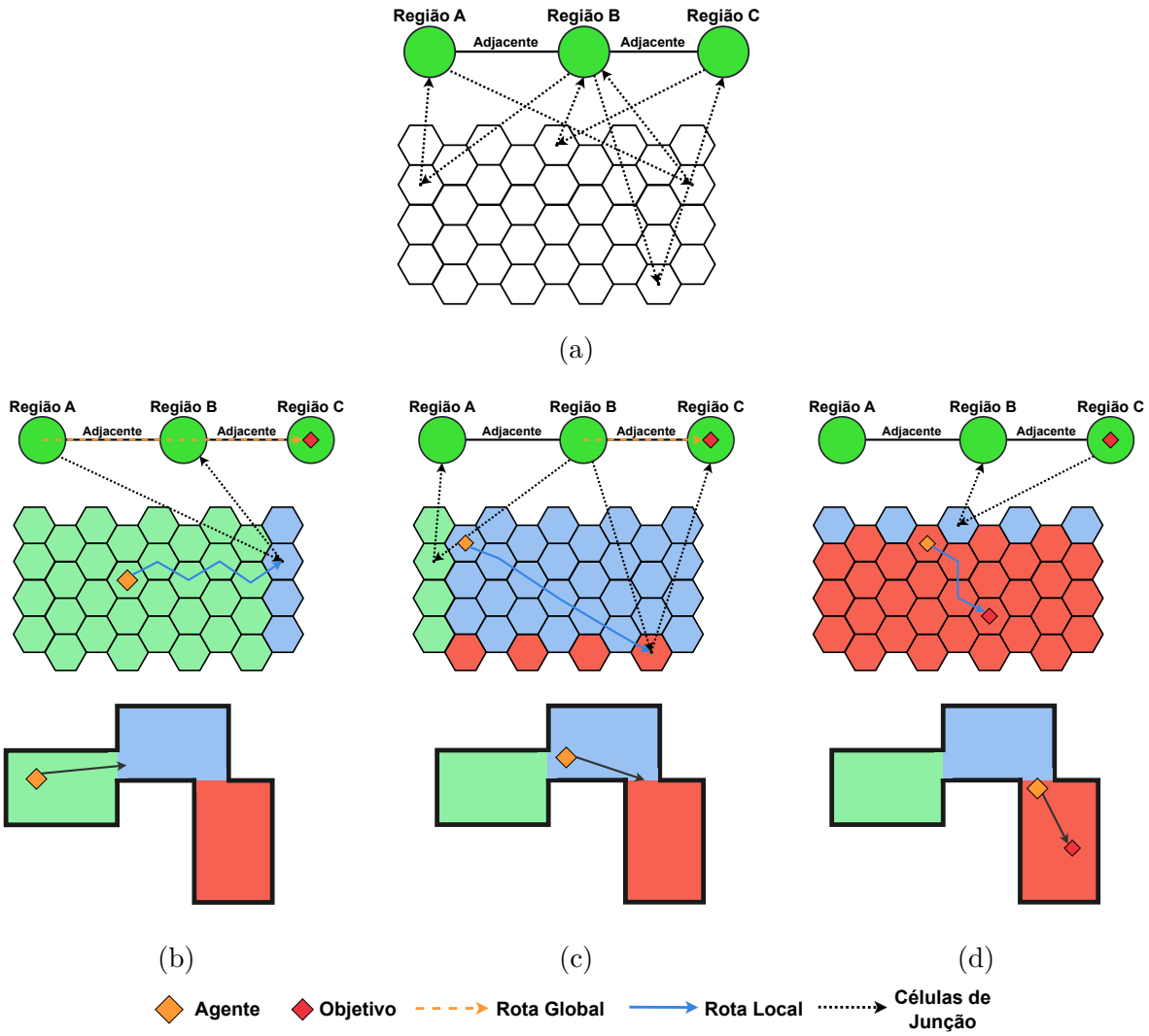


Figura 5.7 – Exemplo de planejamento de rotas realizado pela arquitetura. (a) Representação das regiões conhecidas pelo agente, suas conexões de adjacência e células de junção. As células de junção indicam, para uma dada região, quais células da grade levam a regiões adjacentes e qual é esta região; (b) Planejamento da rota local da região A para B; (c) Planejamento da rota local da região B para C; (d) Planejamento da rota local dentro da região C do agente até o objetivo; (b,c,d) Na parte inferior é mostrado um mapa global do ambiente e a locomoção a ser executada pelo agente correspondente à rota local.

5.6.2 Atuação Motora

Os *codelets* de atuação motora, assim como os *codelets* sensoriais, possuem acesso direto ao ambiente simulado, sendo responsáveis por realizar os comandos correspondentes às ações presentes na memória motora. O ***codelet* de atuação das pernas** lê a memória motora das pernas e envia comandos `moveTo(X, Y)` para o controlador do agente no CoppeliaSim. Enquanto o ***codelet* de atuação das mãos** envia comandos

`eatItThing` para o agente simulado. A funcionalidade desses *codelets* é simples, sendo os mesmos utilizados como um módulo para desacoplar o mecanismo de decisão e planejamento do mecanismo de execução dos atuadores. Com isso, torna-se fácil a inclusão de novos mecanismos de decisão de ação ou novos atuadores ao agente.

5.7 Sistema Episódico

O sistema episódico segue a operação do modelo teórico apresentado na seção 4.3 e seção 4.4, contendo *codelets* de buffer, predição de erro, mudança contextual, formação de episódio, extração de traço de memória e recuperação de episódios. Os *codelets* de buffer possuem funcionamento simples, realizando somente a cópia das *Ideas* presentes em todos os *memory objects* de entrada para saída, garantindo que na memória de saída sempre haja uma lista dos últimos n estados das memórias indexados pelo seus *timestamps*.

A partir do buffer de experiência do agente, formado pelo conjunto do buffer perceptual e cognitivo, o agente realiza a segmentação de eventos e episódios, sua codificação, armazenamento e recuperação. A arquitetura implementada considera dois tipos de contextos na codificação dos episódios: espacial e motivacional. Essas informações são utilizadas para formar o contexto fraco e forte, respectivamente, conforme apresentado na Figura 4.3.

Para implementação desenvolvida aqui, a Figura 5.8 apresenta um exemplo de episódio segundo a representação adotada com contexto espacial e motivacional, onde destaca-se que a mesma trata de uma instanciação do modelo base de representação de episódios da Figura 4.3, sendo que o contexto motivacional é escolhido como contexto forte e o contexto espacial (especialmente a região do agente) é escolhido como contexto fraco. O contexto motivacional indica o impulso ativo do agente durante o evento, indicando o objetivo ativo do agente durante o evento. O contexto espacial refere-se aos objetos identificados pela percepção e a região do agente durante o evento. Este tipo de contexto carrega informações sobre a disposição espacial do ambiente, podendo ser relevante para o planejamento de rotas e simulações mentais do agente. Além disso, a informação das células ocupadas por um objeto são representadas por *links* entre os objetos e *Ideas* que representam cada célula da grade hexagonal separadamente. Isso permite que as *Ideas* de célula de grade sejam conectadas com diferentes objetos.

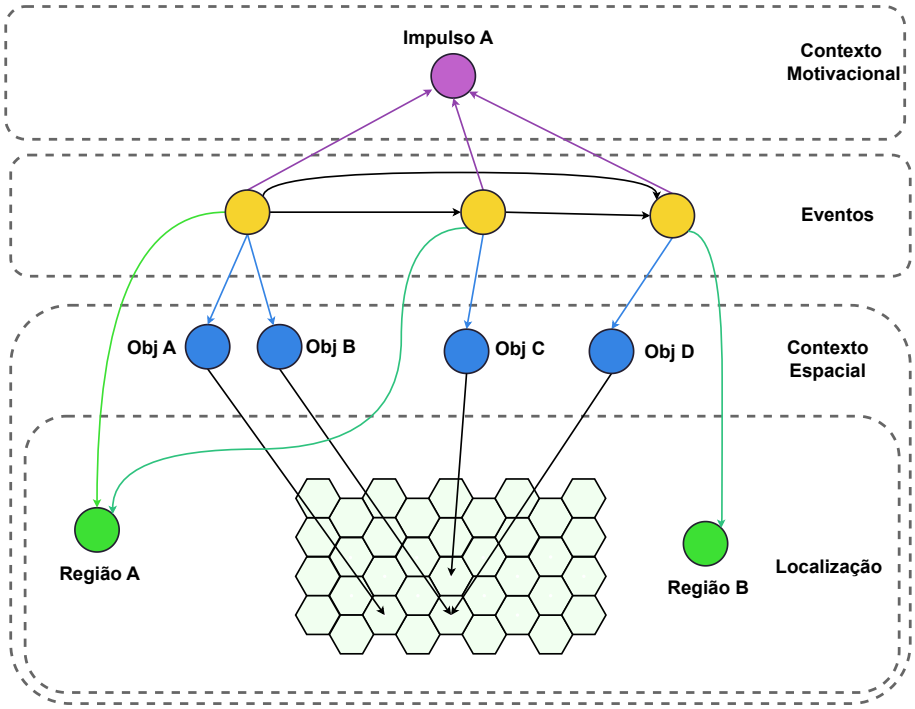


Figura 5.8 – Exemplo de representação de um episódio. Cada círculo e hexágono corresponde a uma *Idea* que contém informação sobre um objeto, evento, região, motivação, ou posição.

5.7.1 Detecção de Eventos

Os ***codelets* de detecção de evento mover, evento rotacionar e evento de coleta** realizam a predição de erro na detecção de eventos, onde cada *codelet* possui uma *Idea* de categoria de evento associada. O funcionamento desses *codelets* segue a descrição apresentada na seção 4.3. Em particular, consideram eventos com transição linear, ou seja, os valores de propriedades observadas devem ter uma mudança aproximadamente linear dentre as amostras observadas. Em todos os casos é utilizada três amostras.

Uma *Idea* de categoria de evento é capaz de detectar se uma sequência qualquer de *Ideas* é uma instância do tipo de evento que ela representa. Para isso, a *Idea* carrega duas informações pertinentes. A primeira é a lista de propriedades relevantes em um objeto. Diferentemente de uma instância de evento, que referencia uma transição específica de um objeto específico, a categoria de um evento representa um tipo de transição que pode ser identificado em objetos que possuem um certo conjunto de propriedades. Dessa forma, a categoria de evento deve verificar se as *Ideas* da sequência a ser analisada possuem as propriedades associadas ao evento. Por exemplo, o evento 'mover' possui como propriedades relevantes as posições X e Y. Com isso, qualquer sequência de *Ideas* que

possuam em sua representação as propriedades X e Y estaria apta a ser classificada como um evento 'mover'.

A segunda informação é uma função para verificação da mudança ocorrida nos valores das propriedades. Esta função recebe uma lista com os valores das propriedades relevantes extraídos das amostras da sequência analisada. A verificação da mudança linear (Figura 5.9) é feita comparando o vetor formado pela diferença entre as amostras 1 e 2 com o vetor de diferenças entre as amostras 2 e 3. A mudança é considerada parte do evento se: a mudança total é maior que um certo limiar; e a diferença de angulação entre os vetores é menor que um certo limiar. O limiar para angulação é mantido em 0.02, enquanto o limiar de mudança total (o qual será referido simplesmente como limiar de detecção de evento) é testado com diferentes valores nos experimentos realizados.

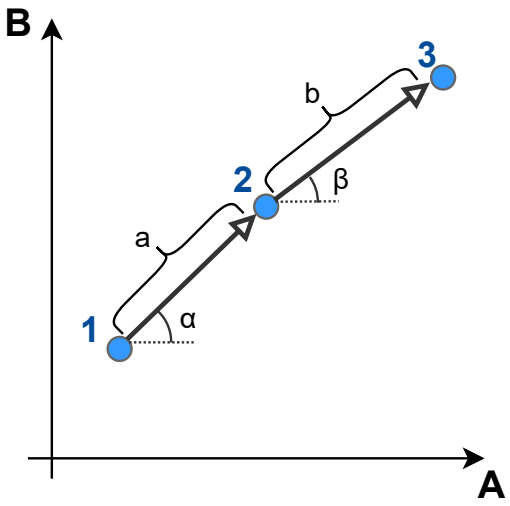


Figura 5.9 – Ilustração da operação de verificação de mudança linear. 1, 2 e 3 são três amostras sequências. Para ser um evento linear, temos que $a + b \geq d$ e $|\alpha - \beta| \leq \phi$, onde d e ϕ são os limiares de comparação.

Dotados de uma *Idea* de categoria de evento, os *codelets* de detecção de evento monitoram o **buffer perceptivo** para identificar na sequência de estados de um objeto quando um evento inicia e termina. Como o buffer perceptivo possui uma sequência dos estados de todas as memórias perceptivas, primeiramente os *codelets* de detecção separam essa sequência para cada objeto que possa ser parte do tipo de evento a ser detectado. Isso é feito pela comparação com a lista de propriedades associada à categoria de evento. Dessa forma, são criadas sequências temporais de estados para cada objeto que são verificadas a cada três amostras pela categoria de evento (Figura 5.10). Quando uma sequência de amostras é identificada como pertencente à categoria de evento, mas a sequência anterior

não era, é detectado o início de um evento. A situação oposta marca o fim do evento. Pode-se pensar na operação dos *codelets* de detecção de evento como *scanners* do fluxo de informação de cada objeto que utilizam a categoria de evento como instrumento para identificar períodos em que a informação era correspondente à categoria.

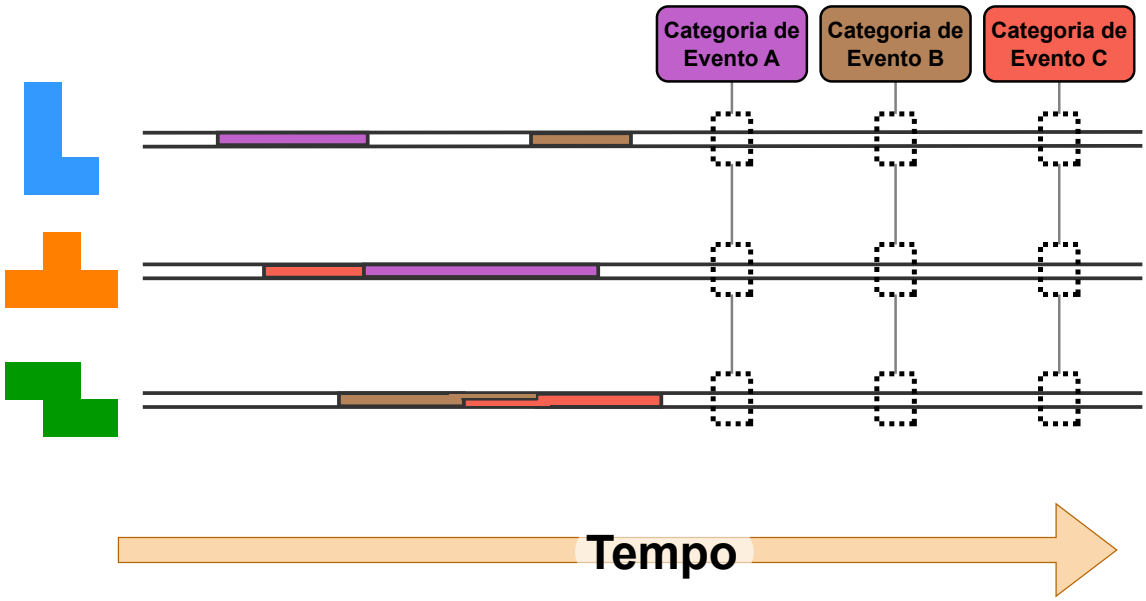


Figura 5.10 – Ilustração do funcionamento dos *codelets* de detecção de evento.

5.7.2 Mudança Contextual

O ***codelet* de mudança contextual** gera as fronteiras fracas e fortes para segmentação dos episódios com base no contexto interno do agente. Fronteiras fracas são detectadas quando há uma mudança de região do agente. Enquanto, fronteiras fortes são detectadas quando há mudança do impulso ativo (i.e., impulso com maior ativação armazenado no *memory container* **impulsos**). A região do agente durante um evento é parte do contexto espacial do mesmo. Dessa forma, caso o agente mude de região durante um evento, é necessário segmentar esse evento em dois eventos, onde cada um tenha somente uma região como contexto espacial (Figura 5.11). Caso não fosse realizada essa segmentação, um evento seria associado a duas regiões distintas e as posições absolutas do agente e demais objetos do contexto espacial não poderiam ser determinadas a partir da célula de grade hexagonal ocupada por cada um. Logo, a fronteira fraca tem o papel de manter a coerência de contexto nos eventos, evitando ambiguidades.

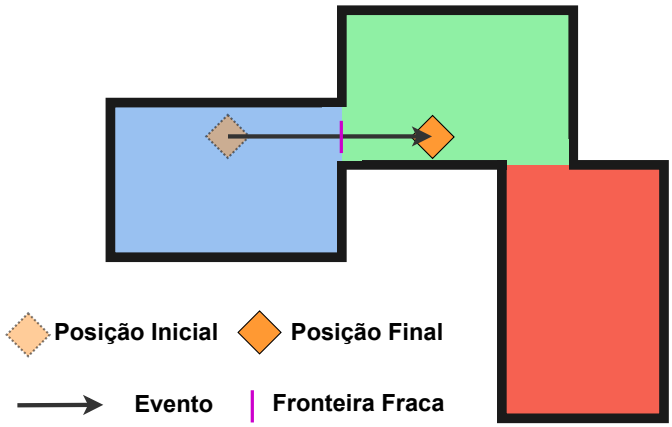


Figura 5.11 – Exemplo de ocorrência de uma fronteira fraca durante um evento de locomoção.

5.7.3 Representação do Episódio

A representação do episódio atual sendo experienciado pelo agente é construída pelo *codelet* **formação de episódio**, conforme detalhado na Seção 4.3 e ilustrado na Figura 4.3. Em específico, o *codelet* segue o Algoritmo 5.1, onde, o contexto contêm a informação da região do agente e o impulso de maior ativação, os perceptos contêm todos os objetos da **memória perceptual de curto-prazo** e o episódio é uma *Idea* de suporte que atua como lista para todas as *Ideas* inseridas na memória. Quando uma *Idea* é inserida na memória, a mesma é encapsulada em uma *Idea* auxiliar chamada "nó". Um nó contém índices para quatro *sub-ideas*:

- **Conteúdo**: carrega a informação sendo adicionada, ou seja, a *Idea* fornecida como parâmetro para operação de inserção;
- **Tipo**: indica o tipo do nó, podendo ser evento, episódio, contexto, posição, ou objeto;
- **Links**: listas indexadas por tipo, indicando as *Ideas* às quais o nó se conecta. Ao todo são utilizados 17 tipos de *links*: **primeiro**, **último** e **próximo** indicam relações entre nós de episódios e evento; **região** e **motivação** conectam eventos aos contextos de região e motivação do agente durante o evento; **contexto espacial**, **início** e **fim** conectam eventos aos *links* espaciais referentes aos estados inicial e final do objeto e a todos os objetos estáticos durante o evento que formam o contexto espacial; **relações de Allen** (das quais são utilizadas somente 7) indicam as relações temporais entre

Algorithm 5.1 Formação de Episódio

Inputs: *eventos*, lista de eventos detectados
contextos, buffer cognitivo
perceptos, buffer perceptivo
episodio, episódio atual
fronteira, o timestep da fronteira forte mais recente
Output: *episodio_finalizado*, se o episódio atual for finalizado ele é retornado
Initialize: $episodio\_finalizado \leftarrow \emptyset$
for all $evento \in eventos$ **do** ▷ Ordenado pelos timestamps
  if $episodio.inicio < fronteira$ **e** $evento.inicio > fronteira$ **then**
    $episodio\_finalizado \leftarrow episodio$
    $episodio \leftarrow$ **new** $Episodio()$
  end if
  if $evento \notin episodio$ **then**
    $contexto \leftarrow contextos[evento.fim]$ ▷ Busca o último contexto antes do fim do evento
    $percepto \leftarrow perceptos[evento.inicio : evento.fim]$
    Insere *evento* no *episodio*
    CRIA-RELAÇÕES-ALLEN(evento, episodio) ▷ Cria as relações de Allen entre o evento novo e todos presentes no episodio
    Insere todos elementos de *contexto* no *episodio*
    Insere em *episodio links* entre *evento* e todos elementos de *contexto*
    for all $objeto \in percepto$ **do**
      if $objeto \neq evento.objeto$ **then**
        Insere *objeto* no *episodio*
        Insere em *episodio link* entre *evento* e *objeto*
      end if
    end for
  end if
end forreturn $episodio\_finalizado$

os intervalos de tempo dos eventos; **objeto** e **posição** conectam um *link* espacial a uma categoria de objeto e posição da célula hexagonal;

- **Backlink**: listas indexadas por tipo indicando as *Ideas* que se conectam ao nó. São criadas automaticamente quando um *link* do nó A para B é adicionado, porém são utilizados somente como auxílio para a execução de algoritmos.

Dessa forma, a *Idea* da memória episódica é tratada como um grafo direcionado de nós e arestas heterogêneas.

Quando um episódio é finalizado, o *codelet* de **extração de traço de episódio** realiza sua integração à **memória episódica de longo-prazo**. Para isso, a representação do episódio é adaptada em uma nova *Idea*, para qual a informação temporal dos eventos

(*timestamps* de início e fim) e os nós de contexto motivacional e região do agente são diretamente copiados. Enquanto, os nós de representações dos objetos do contexto espacial e dos objetos dos estados iniciais e finais dos eventos são transformadas em *links* espaciais.

5.7.3.1 *Links* Espaciais

Links espaciais são *Ideas* que atuam como conectores entre *Ideas* de objetos e uma ou mais *Ideas* de células de grade hexagonal. A representação de objetos no episódio criado pelo *codelet* **formação de episódio** contém todas as propriedades do objeto, incluindo sua posição espacial. O *link* espacial permite a separação da representação da posição espacial do objeto (as células hexagonais ocupadas) e suas demais propriedades. Assim, os estados inicial e final de um evento de deslocamento de um certo objeto pelo ambiente é representado por dois *links* espaciais que referenciam a mesma *Idea* de objeto, porém *Ideas* de célula hexagonal distintas. Semelhantemente, como as *Ideas* de grade de células hexagonais é reaproveitada em cada ambiente, pode haver diferentes objetos representados com a mesma posição na grade, sendo diferenciado pelo contexto da região na qual o agente está presente. Para permitir a reutilização de categorias de objetos e grades de células, os *links* espaciais atuam como *proxies* para uma combinação específica de uma categoria de objeto com uma ou mais células de grade.

Dois processos ocorrem durante a transformação da representação de um objeto em um *link* espacial (Figura 5.12). Primeiro, o objeto é assimilado em uma categoria de objeto (Algoritmo 5.2). Após isso, a combinação da categoria de objeto com a posição espacial do objeto é combinado em um *link* espacial. Caso o *link* espacial já exista na **memória episódica de longo-prazo**, o *link* espacial armazenado é utilizado, caso contrário um novo *link* espacial é criado (Algoritmo 5.3). Assim como as demais informações na *Idea* da memória episódica, os *links* espaciais são armazenados como nós que possuem conexão com os nós de categoria de objeto e posição espacial.

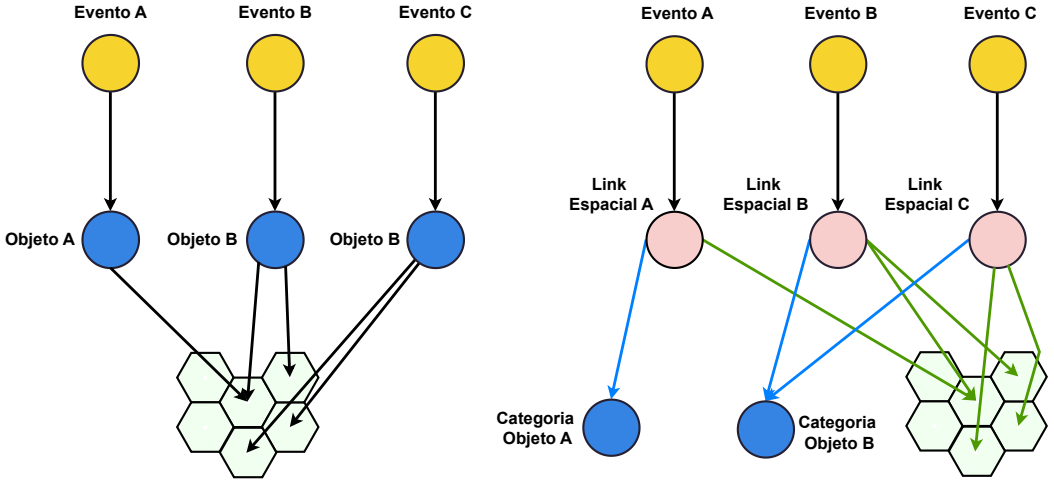


Figura 5.12 – Transformação de um episódio experienciado em um episódio armazenado. Círculos amarelos são *Ideas* de evento, azuis de objetos, rosas de link e hexagonos são *Ideas* de posição. (a) Representação em grafo de um episódio experienciado, i.e., a saída fornecida pelo *codelet* de formação de episódio; (b) Representação em grafo de um episódio armazenado com *links* espaciais.

Algorithm 5.2 Assimilar objeto

Inputs: *objeto*, *Idea* do objeto a ser assimilado
categorias, lista de categorias de objeto existentes
Output: *categoria*, categoria do objeto
function ASSIMILA-OBJETO(*objeto*, *categorias*)
  $melhor\_categoria \leftarrow \emptyset$
  $maior\_valor \leftarrow 0$
  for all $categoria \in categorias$ **do**
    $valor \leftarrow$ TESTA-CATEGORIA($categoria, objeto$)
    if $valor > melhor\_valor$ **then** ▷ Verifica o grau de pertencimento à categoria
      $melhor\_categoria \leftarrow categoria$
      $maior\_valor \leftarrow valor$
    end if
  end for
  if $melhor\_valor > limiar$ **then** ▷ Limiar de decisão $[0, 1]$ para o valor de pertencimento de um objeto a uma categoria
    if $melhor\_valor > (0.5 + limiar/2)$ **then**
      Insere *objeto* como exemplar em $melhor\_categoria$
    end if
    return $melhor\_categoria$
  end if
  $categoria \leftarrow$ **new** $CategoriaObjeto(objeto)$
  return categoria
end function

Algorithm 5.3 Construção de *link* espacial a partir de um objeto

Inputs: *objeto*, instância de objeto a ser transformado em *link* espacial
memoria, memória episódica de longo prazo
Output: *link_espacial*, *link espacial* referente ao objeto
function CONSTRUIR-LINK-ESPACIAL(objeto, memória)
posicao ← *objeto.posicao*
Remove a informação *posicao* de *objeto*
categoria ← ASSIMILA-OBJETO(*objeto*, *memoria.categorias_objetos*)
link_espacial ← *memoria.pai*(*categoria*, *posicao*) ▷ Retorna, se existente, o nó pai em comum entre os itens.
if *link_espacial* = ∅ **then**
link_espacial ← **new** *LinkEspacial*(*categoria*, *posicao*)
Insere *link_espacial* em *memoria*
end if
return *link_espacial*
end function

5.7.3.2 Categorias de Objetos

As categorias de objetos são utilizadas para definir conjuntos de objetos semelhantes. Mais especificamente, uma categoria de objetos refere-se a diferentes observações de um mesmo objeto, excluindo sua posição espacial (a posição espacial é incorporada no *link* espacial). Pode-se adicionar novos exemplares ao conjunto da categoria, permitindo sua adaptação conforme novos episódios são incorporados. Cada categoria de objetos carrega a informação sobre o ID do objeto a que se refere (o ID é atribuído pela simulação e é único para cada objeto existente), uma lista de propriedades e uma lista de exemplares. A lista de propriedades, assim como o ID, é obtida do primeiro exemplar adicionado à categoria, consistindo de todas as propriedades que carregam uma informação numérica (posição, cor, angulação, etc.). A lista de exemplares é uma lista de listas contendo somente os valores das propriedades extraídos dos exemplares adicionados à categoria.

O teste para verificar que uma dada *Idea* de objeto pertence a uma dada categoria de objetos (Algoritmo 5.4) consiste em verificar se o objeto testado é uma instância do objeto à qual a categoria se refere, comparando a igualdade do ID do objeto ao da categoria, e caso verdadeiro, retornando um valor inversamente proporcional à diferença dos valores das propriedades do objeto testado e o exemplar mais semelhante existente na categoria. Essa computação é realizada considerando cada lista de valores de propriedade como um ponto em dimensão n, onde n é o tamanho da lista de propriedades e computando as diferenças como distâncias entre pontos. Particularmente, mantêm-se uma

árvore KD (PROCOPIUC *et al.*, 2003) para cada categoria de objetos contendo os pontos de cada exemplar. Assim, o exemplar mais semelhante ao objeto testado é obtido por uma busca nesta árvore. Complementarmente, a inserção de novos exemplares à categoria consiste em extrair os valores das propriedades em uma lista e inseri-la na árvore.

Algorithm 5.4 Função de pertencimento a uma categoria de objetos

Inputs: *categoria*, a categoria de objeto a qual será comparada
objeto, a instância de objeto a ser testada
function TESTA-CATEGORIA(categoria, objeto)
  if $categoria.id == objeto.id$ **then**
    Extrai os valores das propriedades relevantes de $objeto \rightarrow propriedades$
    Realiza busca pelo exemplar mais próximo de $propriedades \rightarrow similar$
    return $e\hat{} - [\text{DISTANCIA}(similar, propriedades)/(\sqrt{similar.len})]^2$
  end if
  return 0
end function

5.7.4 Incorporação de Novos Episódios

Após identificar o *link* espacial referente a cada objeto do episódio, os mesmos são adicionados à memória de longo-prazo, caso sejam novos (valor de novidade maior ou igual a 0.9), e conectados aos eventos que os referenciam. O teste do valor de novidade dos objetos de contexto faz o papel de um filtro de atenção, permitindo o armazenamento não atualizados a algum tempo. Adicionalmente, é criada uma *Idea* para referenciar o conjunto de eventos do novo episódio como uma unidade, quando incorporada à memória. Essa *Idea* recebe um ID para identificá-la e índices para o primeiro e último evento do episódio (considerando ordem cronológica). Por fim, o episódio anterior ao novo recebe um *link* para o novo episódio, indicando sua sequência.

5.7.5 Recuperação de Episódios

O *codelet* **recuperação de episódios** realiza a busca na memória episódica de longo-prazo, quando uma *Idea* contendo uma representação parcial de um episódio é inserida no *memory object* **pista**. A *Idea* do episódio parcial (pista) para realizar a busca pode conter diferentes elementos, dentre eles:

- **Eventos**: representações de eventos, contendo sua categoria e/ou *links* para os estados iniciais e/ou finais dos objetos;

Algorithm 5.5 Incorporação de novos episódios

Inputs: *episodio*, *Idea* do episódio a ser incorporado representado em grafo
memoria, grafo da memória episódica de longo prazo
function EXTRACAO-TRACO(episodio, memoria)
for all $evento \in episodio$ **do**
$evento\_lp \leftarrow$ **new** $Evento()$
Copia os timestapms de *evento* em *evento_lp*
Insere *evento_lp* em *memoria*
$inicio\_link\_espacial \leftarrow$ CONSTRUIR-LINK-ESPACIAL$(evento.inicio, memoria)$
Insere *link* de início entre *evento_lp* e *inicio_link_espacial* em *memoria*
$fim\_link\_espacial \leftarrow$ CONSTRUIR-LINK-ESPACIAL$(evento.fim, memoria)$
Insere *link* de fim entre *evento_lp* e $fim\_link\_espacial$ em *memoria*
for all $objeto \in episodio$ com *link* de contexto espacial de *evento* **do**
if Valor de novidade de $objeto \geq 0.9$ **then**
$obj\_link\_espacial \leftarrow$ CONSTRUIR-LINK-ESPACIAL$(objeto, memoria)$
Insere *obj_link_espacial* em *memoria*
Insere *link* de contexto espacial entre *evento_lp* e *obj_link_espacial* em *memoria*
end if
end for
memoria.**insere**(*evento_lp*)
end for
episodio_idea $\leftarrow$ **new** $Idea()$
Insere *episodio_idea* em *memoria*
Insere *link* de primeiro entre *episodio* e o primeiro *evento_lp* em *memoria*
memoria.ultimo_episodio.proximo $\leftarrow$ *episodio_idea*
memoria.ultimo_episodio $\leftarrow$ *episodio_idea*
return memória
end function

- **Relações temporais**: *links* de relações temporais entre eventos. As relações temporais dos eventos em memória são computadas durante a busca, além das relações já existentes, dessa forma, é possível encontrar relações entre eventos de episódios distintos (cujas relações temporais não foram criadas durante o processo de formação de episódio);

- **Objetos de contexto**: objetos de contexto espacial com *links* para suas células de grade;

- **Episódio**: pode-se buscar um episódio específico somente com seu ID (caso esta informação já seja conhecida, por exemplo o ID de episódios recuperados em instâncias passadas e armazenados em outra memória), ou um episódio com *links* para o primeiro e último evento. Esses eventos podem ter qualquer uma das informações

listadas anteriormente.

O Algoritmo 5.6 é utilizado para a recuperação de episódios da memória episódica de longo-prazo. Para tal pode ser necessário realizar uma busca por subgrafos isomorfos à pista. Isso é feito por uma implementação simplificada do processo de refinamento proposto em Ullmann (1976) e também explicado em Bonnici *et al.* (2013), onde é construída uma árvore de busca com a informação a priori dos nós válidos da memória para cada nó da pista. A informação a priori é obtida do fato que, por exemplo, nós de evento da pista só podem ser mapeados em nós de evento da memória episódica. Durante a busca, cada nó folha da árvore de busca corresponde a um possível mapeamento da pista para um subgrafo da memória. Quando o mapeamento possuir alguma combinação que não respeita a estrutura da pista, é realizado o corte dos caminhos da árvore de busca que teriam a mesma combinação inválida, reduzindo assim o espaço de busca.

Ao fim da recuperação, os episódios encontrados são convertidos para a representação de episódios gerada pelo *codelet* de formação de episódios. Ou seja, o processo da Figura 5.12 é revertido, transformando as *Ideas* dos *links* espaciais em *Ideas* de objetos. Neste processo, é construída uma *Idea* com os valores de propriedades de um exemplar amostrado aleatoriamente da categoria de objeto referenciada pelo *link* espacial e a posição referenciada pelo mesmo *link* espacial.

Algorithm 5.6 Recuperação de episódios em memória a partir de uma pista.

Inputs: *pista*, a *Idea* de um episódio parcial
memoria, grafo da memória episódica de longo prazo
function RECUPERA-EPISODIOS(pista, memoria)
recuperados $\leftarrow \emptyset$
eventos_validos = *memoria.eventos*
if *pista.contexto_espacial* $\neq \emptyset$ **then**
Remove os eventos de *eventos_validos* que não possuem *link* espacial para ao menos um objeto de *pista.contexto_espacial*
end if
if *pista.eventos* $\neq \emptyset$ **then**
mapa_eventos $\leftarrow \{\}$
for all *evento* $\in$ *pista.eventos* **do**
mapa_eventos[*evento*] $\leftarrow$ *eventos_validos* que sejam da mesma categoria que *evento*
if *evento.inicio* $\neq \emptyset$ **and** $evento.fim \neq \emptyset$ **then** ▷ Evento da pista define um estado inicial e final de objeto?
mapa_eventos[*evento*] $\leftarrow$ *mapa_eventos*[*evento*] que iniciam e terminam com *links* espaciais correspondentes aos objetos *evento.inicio* e $evento.fim$
end if
end for
end if
if *mapa_eventos*[*evento*] $= \emptyset, \forall evento \in$ *pista.eventos* **then**
if *pista.contexto_espacial* $\neq \emptyset$ **then**
recuperados $\leftarrow$ todos episódios na *memoria* contendo ao menos um dos eventos em *eventos_validos*
end if
else if size(*pista.eventos*) $= 1$ **then**
recuperados $\leftarrow$ todos episódios na *memoria* contendo ao menos um dos eventos em *mapa_eventos*[*pista.eventos*[0]]
else if size(*pista.eventos*) > 1 **then**
validos $\leftarrow$ busca por subgrafos em *memoria* isomorfos a *pista* considerando as correspondências válidas de *mapa_eventos*
menor $\leftarrow$ o subgrafo com menor intervalo de tempo (início do evento mais antigo até fim do evento mais recente)
recuperados $\leftarrow$ todos episódios na *memoria* contendo ao menos um dos eventos do subgrafo *menor*
end if
Converte os *links* espaciais em *recuperados* para objetos
return *recuperados*
end function

6 Experimentos e Resultados

6.1 Metodologia

A validação do modelo proposto e implementado foi feita através de experimentos simulados controlados para observação do comportamento dos módulos desenvolvidos. Para cada experimento busca-se avaliar se o comportamento observado do módulo de memória episódica corresponde ao esperado conforme as propostas de sua descrição. Além disso, os experimentos permitem a exploração da variação de parâmetros e discussões qualitativas do modelo proposto.

6.2 Experimento 1: Segmentação de Eventos e Reutilização de Elementos de Memória

Este experimento busca avaliar a correta segmentação de eventos pelo módulo de detecção de eventos, assim como a capacidade do mesmo de reutilizar os elementos da memória de longo prazo para integração de novos episódios. O cenário de simulação consiste em um ambiente simples no qual um agente, referido aqui como **agente episódico**, controlado pela arquitetura implementada é posicionado, conforme mostra a Figura 6.1. A posição do agente episódico é mantida fixa e dois outros agentes sem capacidade de memória episódica, referidos aqui como **atores**, executam uma sequência de movimentos predeterminados[1] que devem ser observados, modelados e representados pelo agente episódico. O agente episódico é imbuído de um mapa do ambiente, que consiste em apenas uma região. Além disso, os módulos motivacionais e de atuação são removidos, impedindo que o agente episódico realize ações. Os atores e agente episódico possuem nível de energia fixos, não sendo necessária a coleta de alimentos para sua reposição.

Os atores executam uma mesma sequência de movimentos quatro vezes enquanto o agente episódico observa a cena, codifica os episódios e os armazena na memória. No final da execução de cada sequência de movimentos dos atores, o agente episódico é forçado a segmentar o episódio. Para isso, uma fronteira forte é inserida diretamente

[1] Um exemplo de execução da simulação pode ser visto em <https://youtu.be/YlbGC3R9UhA>

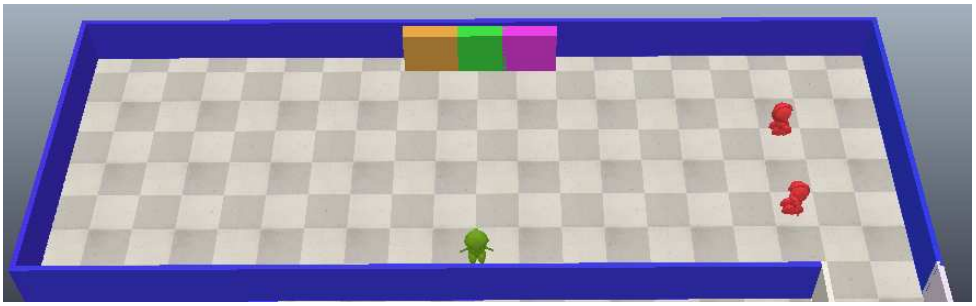


Figura 6.1 – Cenário de simulação simples. O agente em verde (agente episódico) é controlado pela arquitetura implementada. Os agentes em vermelho (atores) executam movimentos pré-determinados.

no *memory object* de fronteiras de episódios do agente episódico. Isso é necessário, pois nenhum dos módulos motivacionais foi instanciado nesse experimento, logo o agente episódico não possui informações no *buffer* cognitivo para detectar transições de episódios.

No total este cenário de simulação simples foi executado 78 vezes, modificando os limiares de detecção de evento e pertencimento à categoria de objetos (Algoritmo 5.2) em cada execução. O limiar de detecção de evento (denominado na sequência somente como limiar de evento), ilustrado na Figura 5.9, determina o valor mínimo da diferença entre amostras de um objeto que pode ser caracterizado como um evento para todas as categorias de evento do agente episódico. O limiar de pertencimento à categoria de objetos (denominado a partir daqui somente como limiar de categoria de objeto) é o valor mínimo utilizado para decidir se uma instância de objeto pertence ou não a uma categoria de objeto com base no valor retornado pela função de pertencimento da categoria (vide Algoritmos 5.2 e 5.4). Dessa forma, o limiar de categoria de objeto regula o quão "generalizadas" as categorias de objetos serão, onde valores maiores levam a categorias mais específicas e com poucos exemplares, enquanto valores menores levam a categorias mais amplas e com vários exemplares.

Em cada execução, os atores executam uma mesma sequência de movimentos 4 vezes e 4 episódios são armazenados pelo agente episódico. Adicionalmente, a cor dos atores altera ligeiramente ao longo do tempo, proporcionando variações a serem consideradas na criação de categorias de objeto.

6.2.1 Resultados e Discussões

A Tabela 6.1 apresenta a quantidade total de eventos armazenados para cada execução, enquanto a Tabela 6.2 apresenta a quantidade de categorias de objetos criadas.

	Limiar de Evento					
Limiar de Objeto	**0.5**	**0.25**	**0.1**	**0.01**	**0.001**	**0.0001**
1.0	0	8	33	34	34	53
0.99	0	13	33	36	34	46
0.95	0	14	33	34	34	49
0.9	0	15	33	35	34	47
0.8	0	9	34	35	35	47
0.7	0	10	34	34	34	46
0.6	0	15	33	34	34	48
0.5	0	16	33	36	35	51
0.4	0	19	33	34	26	50
0.3	0	10	33	26	34	51
0.2	0	9	33	34	34	45
0.1	0	15	33	34	35	46
0.0	0	17	25	34	35	47

Tabela 6.1 – Número de eventos armazenados ao final da simulação para diferentes valores de limiar de detecção de evento e de categoria de objeto.

	Limiar de Evento					
Limiar de Objeto	**0.5**	**0.25**	**0.1**	**0.01**	**0.001**	**0.0001**
1.0	0	17	70	72	72	97
0.99	0	11	42	50	47	45
0.95	0	12	27	24	25	27
0.9	0	11	23	27	22	25
0.8	0	7	20	22	25	22
0.7	0	8	18	17	17	20
0.6	0	8	15	17	17	18
0.5	0	9	14	15	15	17
0.4	0	13	13	14	14	15
0.3	0	4	10	13	13	13
0.2	0	3	10	12	13	12
0.1	0	4	9	11	11	11
0.0	0	4	7	9	9	9

Tabela 6.2 – Número de categorias de objetos criadas ao final da simulação para diferentes valores de limiar de detecção de evento e de categoria de objeto.

Observa-se na Tabela 6.1 uma faixa ampla de limiares de detecção de evento (de 0.1 a 0.001) para os quais a quantidade de eventos detectados praticamente não se altera. Para um conjunto semelhante de observações, a arquitetura é capaz de consistentemente integrar as variações de propriedades observadas em eventos. Nota-se também que, ao diminuir o limiar de detecção de evento para 0.0001 o número de eventos aumenta. O software CoppeliaSim, utilizado nas simulações, fornece leituras ruidosas dos sensores simulados, pois o mesmo busca reproduzir situações realistas da operação de robôs. Dessa

forma, para valores muito pequenos do limiar de detecção de evento, a arquitetura passa a detectar esse ruído de baixa amplitude presente na detecção dos sensores da simulação como sendo eventos do objeto. Esse comportamento não é desejado, pois gera eventos em excesso que não carregam informação significativa. Similarmente, ao aumentar o limiar para 0.5 o número de eventos é nulo, pois as variações observadas são menores que o limiar, impedindo a detecção dos eventos.

Embora esse comportamento não seja desejado na escala espaço-temporal utilizada no experimento, pode-se imaginar uma variação do sistema episódico no qual múltiplas escalas de tempo são consideradas, cada uma com um limiar de detecção diferente. Isso permitiria a detecção de eventos que ocorrem devido a processos mais lentos. Adicionalmente, poderíamos realimentar o sistema de forma que episódios relembrados fossem armazenados em um *buffer* para detecção de eventos em escala temporal grande, mesmo que os estados intermediários não tenham sido detectados. Assim, um agente poderia detectar possíveis eventos que ocorreram entre observações temporalmente distintas uma das outras sem a necessidade de um aparato específico para detectar este tipo de evento.

Observa-se também, que o limiar de pertencimento à categoria de objeto afeta, como esperado, o número de categorias criadas. Para o valor 1.0, o reconhecimento da categoria ocorre somente para correspondências exatas dos valores de propriedades do objeto testado e do exemplar armazenado na categoria. Conforme esse valor é decrementado, folgas maiores são permitidas nas diferenças de valores entre o objeto testado e os exemplares, necessitando de menos categorias para identificar todos os objetos. No extremo, onde o limiar é 0.0, o critério de comparação entre os valores de propriedades é irrelevante e as categorias de objetos garantem somente que o objeto testado possua o mesmo ID do exemplar.

Semelhantemente às variações do limiar de detecção de evento, pode-se considerar uma alternativa interessante da implementação da arquitetura na qual diferentes valores de limiares são utilizados para realizar uma organização hierárquica das categorias de objetos. Dessa forma, categorias mais "gerais" seriam avaliadas com limiares menores e categorias mais "específicas" possuiriam limiares maiores. Isso permitiria a adição de mecanismos que abstraíssem mais (utilizando categorias mais gerais) episódios pouco "relevantes", enquanto episódios mais "relevantes" poderiam ter categorias mais específicas. Com isso, pode-se controlar o nível de fidelidade da reconstrução dos episódios.

Na Figura 6.2 é detalhado o uso dos links espaciais na memória episódica para a execução com limiares de evento e de categoria de objeto definidos em 0.01 e 0.6 respectivamente. O gráfico da Figura 6.2a apresenta, na forma de uma matriz esparsa, quais os links espaciais conectados em cada um dos eventos armazenados na memória de longo-prazo. Assim, pode-se interpretar o gráfico como uma visualização da matriz de adjacência entre os nós de eventos e nós de links espaciais na memória episódica de longo-prazo. Os eventos são ordenados cronologicamente com base em sua ordem de detecção e são indexados ao longo do eixo x do gráfico, ou seja, os valores apresentados no eixo x são índices numéricos criados em pós-processamento dos dados da simulação de forma a ordenar os eventos criados, não são representados os instantes temporais de inicio e término dos eventos. Para cada evento (cada coluna do gráfico) é apresentado o índice do link espacial que representa o estado inicial do objeto em azul, o link espacial do estado final do objeto em vermelho e os links espaciais que representam o contexto espacial do evento em verde. Por exemplo, o evento 1 (primeiro evento na Figura 6.2a mais a esquerda) tem o link espacial 0 como estado inicial, o link espacial 1 como estado final e o link espacial 2 como contexto espacial. Enquanto que o evento 2, reutiliza o link espacial 1 como seu estado inicial, acrescenta o link espacial 3 como estado final e os links espaciais 4, 5, 6 e 7 como contexto espacial.

Um link espacial é conectado a uma categoria de objeto e uma ou mais células de grade que indicam a posição espacial do objeto representado pelo link espacial. Assim, pode-se criar outros dois gráficos que mostram a relação dos eventos com as informações de categoria de objeto e posição espacial separadamente. A Figura 6.2b apresenta a relação entre nós de eventos e os nós de categoria de objetos referenciados pelos *links* espaciais conectados aos eventos, enquanto a Figura 6.2c apresenta a relação com os nós de posição espacial referenciados pelos *links* espaciais. A Seção 7.1 apresenta gráficos adicionais para os demais valores do limiar de categoria de objeto.

Observa-se que somente para os dois primeiros eventos são criados *links* do tipo contexto espacial para *links* espaciais de objetos do ambiente. Isso se deve ao fato de que o agente episódico permanece parado durante todo o experimento. Dessa forma, as paredes e objetos do cenário (objetos nas cores laranja, verde, magenta e azul na Figura 6.1) estão sempre no campo de visão do agente episódico e por consequência o valor de novidade correspondente a eles não é incrementado após o início da simulação. Com isso, os nós de

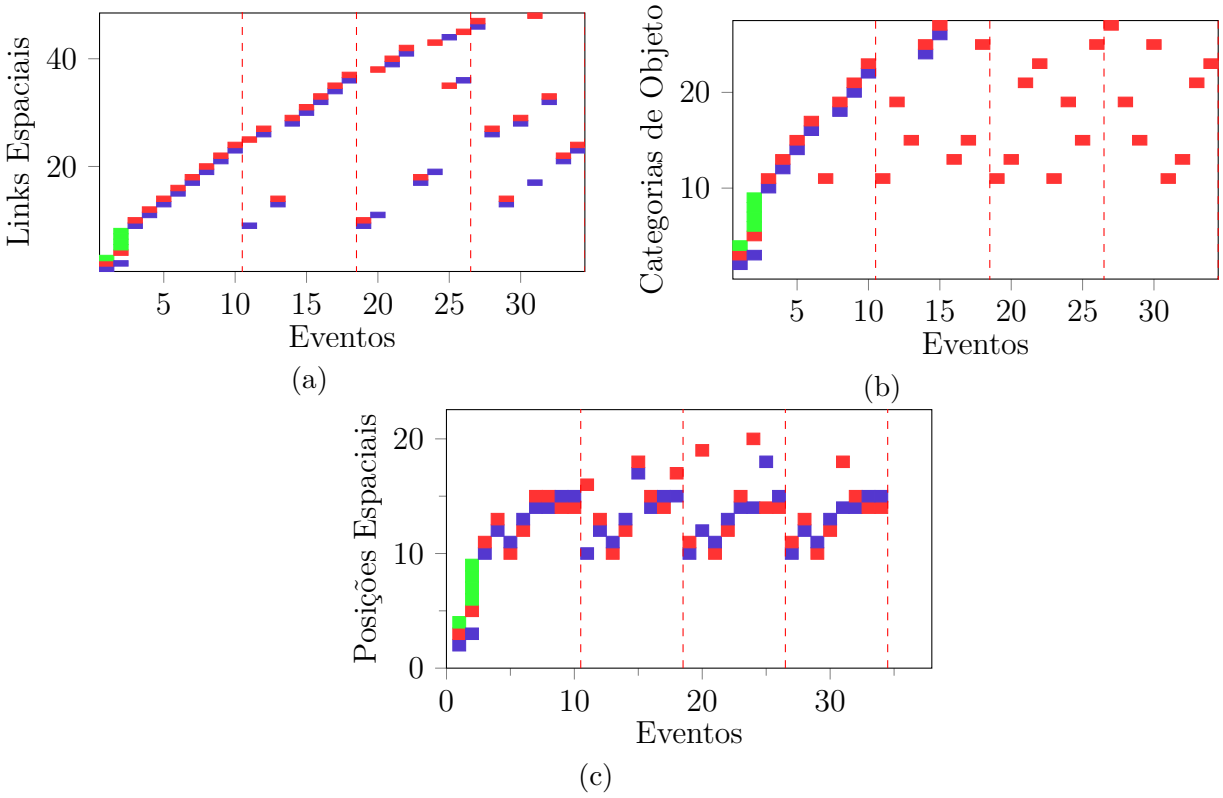


Figura 6.2 – Relação entre nós de eventos e as informações dos links espaciais conectados a eles.(a) Links espaciais conectados aos eventos na memória episódica; (b) Categorias de objetos referidos pelos links espaciais do evento; (c) Células de grade referidas pelos links espaciais do evento. Verdes indicam uma conexão do tipo contexto espacial. Azuis indicam uma conexão do tipo início de evento. Vermelhos indicam uma conexão do tipo final de evento. Linhas tracejadas vermelhas indicam os instantes de segmentação de episódios.

contexto espacial referentes aos mesmos não são armazenados na memória de longo prazo após os primeiros eventos.

É possível notar o reuso das categorias de objetos e posições espaciais para representação de links espaciais. Além disso, alguns dos links espaciais também são reutilizados para representação de novos eventos, conforme o comportamento esperado. O gráfico 6.2c permite também observar uma repetição quase exata das posições dos atores em cada episódio. No extremo do limiar de categoria de objeto (Figura 7.14), os episódios formados permitem detectar claramente a repetição de eventos que está ocorrendo no ambiente.

6.3 Experimento 2: Recuperação de Episódios

Este experimento busca avaliar a recuperação de episódios. O ambiente de simulação é composto por oito salas e três corredores. Em cada sala, há três joias e dois

alimentos. Quatro atores são posicionados no ambiente. Os atores não possuem sistema episódico e rotas pré-determinadas são selecionadas para eles executarem. Cada rota começa na extremidade de um dos corredores (as bordas do ambiente), passa por uma sala e termina na extremidade de um corredor. Os locais de início da rota, final da rota e a sala visitada são selecionados aleatoriamente entre as opções existentes. Quando o ator entra em uma sala, o mesmo verifica se há uma joia no seu campo de visão. Se houver, ele se desloca até ela e a coleta, inserindo-a em seu inventário. O agente episódico tem a capacidade de registrar tal ação através do ***codelet* de detecção de evento de coleta**.

A Figura 6.3 mostra o ambiente simulado[2]. As cores indicam as 11 regiões nas quais o ambiente é subdividido para a criação do mapa interno do agente episódico. O agente episódico é inicializado na região A com os módulos de motivação e atuação e executa-se a simulação por 5 minutos. Durante a simulação, o ***codelet* de impulso de exploração** faz com que o agente episódico explore o ambiente de forma aleatória e observe vários eventos realizados pelos atores. Todos os episódios detectados são incorporados à memória episódica de longo prazo.

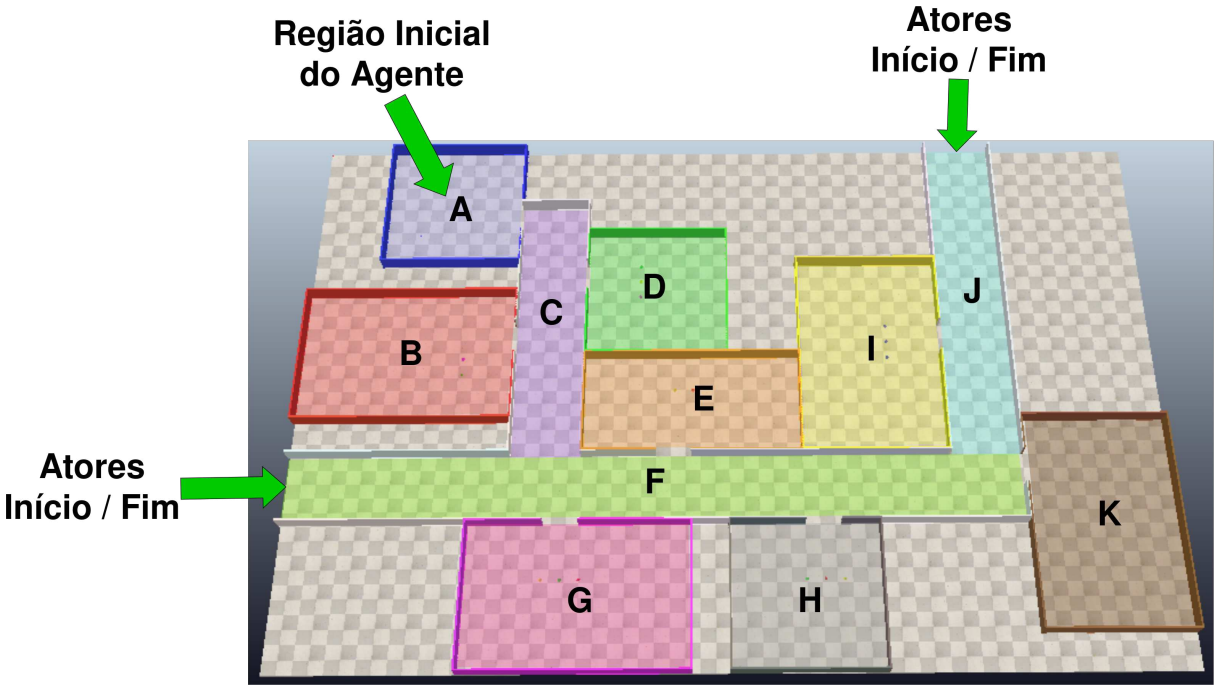


Figura 6.3 – Cenário de simulação amplo. Cada sobra retangular colorida delimita uma região utilizada para construir o mapa interno do agente episódico.

[2] Um exemplo de execução da simulação pode ser visto em <https://youtu.be/eGlsH6M4P7w>

6.3.1 Resultados

O experimento é executado 5 vezes com limiar de detecção de evento 0.01 e pertencimento à categoria de objeto 0.7. A Tabela 6.3 apresenta, para cada execução, o número total de eventos e episódios detectados, o total de instâncias de objetos codificados ao longo da execução e o total de links espaciais armazenados na memória ao final da execução.

	Execução				
	1	2	3	4	5
Eventos Detectados	175	151	161	189	232
Episódios Detectados	8	9	8	11	7
Objetos Codificados	3258	2614	2669	2790	2721
Links Espaciais Criados	309	278	271	286	366

Tabela 6.3 – Total de eventos e episódios criados em cada execução do experimento 2.

Para cada um dos atores, foi criada uma pista para busca na memória episódica. A pista consiste de um nó evento sem categoria definida e dois nós de objeto contendo somente a informação do ID do objeto do ator. Com isso, busca-se recuperar da memória todos os episódios que contêm ao menos um evento com o ator. A Tabela 6.4 apresenta, para cada ator, o índice dos episódios codificado que possuem ao menos um evento envolvendo o ator (i.e., representando uma mudança nas propriedades do objeto do ator) e o índice dos episódios recuperados pela pista. Observa-se que a arquitetura obtêm 100% de sucesso na recuperação, demonstrando a correta operação do módulo de recuperação.

A Tabela 6.3 também permite observar qualitativamente o nível de compressão de informação que ocorre durante o armazenamento de episódios. Para todas as execuções, a quantidade de nós de objetos criados durante as codificações dos episódios é muito maior que a quantidade final de links espaciais que são criados para representar esses objetos na memória de longo prazo. Grande parte desse fator deve-se ao filtro de objetos contextuais com base no seu valor de novidade (Algoritmo 5.5), mostrando a importância da presença de algum filtro atencional/importância no momento de armazenamento para não sobrecarregar a memória de longo prazo.

Execução	Ator	Codificados	Recuperados
1	1	0 - 2	0 - 2
	2	0 - 1 - 2 - 5 - 6 - 7	0 - 1 - 2 - 5 - 6 - 7
	3	2 - 5	2 - 5
	4	2 - 6	2 - 6
	5	2 - 5	2 - 5
2	1	0 - 2	0 - 2
	2	0 - 1 - 6	0 - 1 - 6
	3	0 - 1 - 6 - 8	0 - 1 - 6 - 8
	4	0 - 1 - 6 - 8	0 - 1 - 6 - 8
	5	6	6
3	1	0 - 1 - 5 - 6	0 - 1 - 5 - 6
	2	0 - 1 - 5 - 6	0 - 1 - 5 - 6
	3	0 - 1 - 5	0 - 1 - 5
	4	0 - 1 - 6	0 - 1 - 6
	5	1 - 6	1 - 6
4	1	1 - 9	1 - 9
	2	0 - 1 - 3 - 4 - 7	0 - 1 - 3 - 4 - 7
	3	1 - 9	1 - 9
	4	1	1
	5	8	8
5	1	2 - 3	2 - 3
	2	2 - 3 - 4 - 5	2 - 3 - 4 - 5
	3	3	3
	4	2 - 3 - 4 - 5	2 - 3 - 4 - 5
	5	0	0

Tabela 6.4 – Índices dos episódios codificados e recuperados para cada ator em cada execução.

6.4 Experimento 3: Análise Qualitativa dos Episódios Armazenados

Neste experimento, o cenário de simulação do experimento 2 é repetido, porém não é imposto um tempo limite de execução de 5 minutos. A simulação é executada até que o agente episódico fique sem energia para continuar sua operação. Busca-se com isso avaliar qualitativamente as características dos episódios armazenados na memória episódica.

6.4.1 Resultados

O experimento foi realizado 5 vezes e a Tabela 6.5 apresenta o número de eventos, episódios e links espaciais registrados em cada simulação, além do tempo total de simulação. Em todas as execuções o limiar de detecção de evento é mantido em 0.01 e o de categoria de objeto em 0.5.

	Execução				
	1	2	3	4	5
Eventos Detectados	955	1094	1108	1038	960
Episódios Detectados	55	63	63	56	56
Links Espaciais Criados	1454	1589	1646	1480	1413
Tempo	32min	34min	33min	31min	28min

Tabela 6.5 – Total de eventos, episódios e *links* espaciais criados em cada execução do experimento 3. O tempo total de execução é mostrado desconsiderando os segundos.

Observa-se que o agente episódico registra consistentemente uma média de aproximadamente 31 eventos por minuto. Mesmo sendo um ambiente dinâmico no qual eventos podem variar sua duração, o número limitado de rotas que pode ser tomada (sempre indo de uma região atual até a região a ser explorada) cria uma consistência no número médio de eventos gerados que é refletida na memória episódica do agente episódico.

A Figura 7.19 apresenta as relações entre os eventos detectados e os links espaciais referentes a cada um, assim como as categorias de objeto e a região na qual o agente episódico estava presente durante cada evento. Observa-se que, diferentemente dos gráficos resultantes do experimento 2, há uma presença maior de *links* espaciais conectados aos eventos como contexto espacial (pontos verdes). Como o agente episódico fica períodos sem visualizar alguns dos objetos do ambiente, ao revisitá-los seus níveis de novidade são incrementados significantemente, fazendo com que sejam incorporados na memória de longo prazo. Esse efeito não ocorria no experimento 2, pois o agente episódico permanecia estático, logo o nível de novidade dos objetos estáticos não era incrementado.

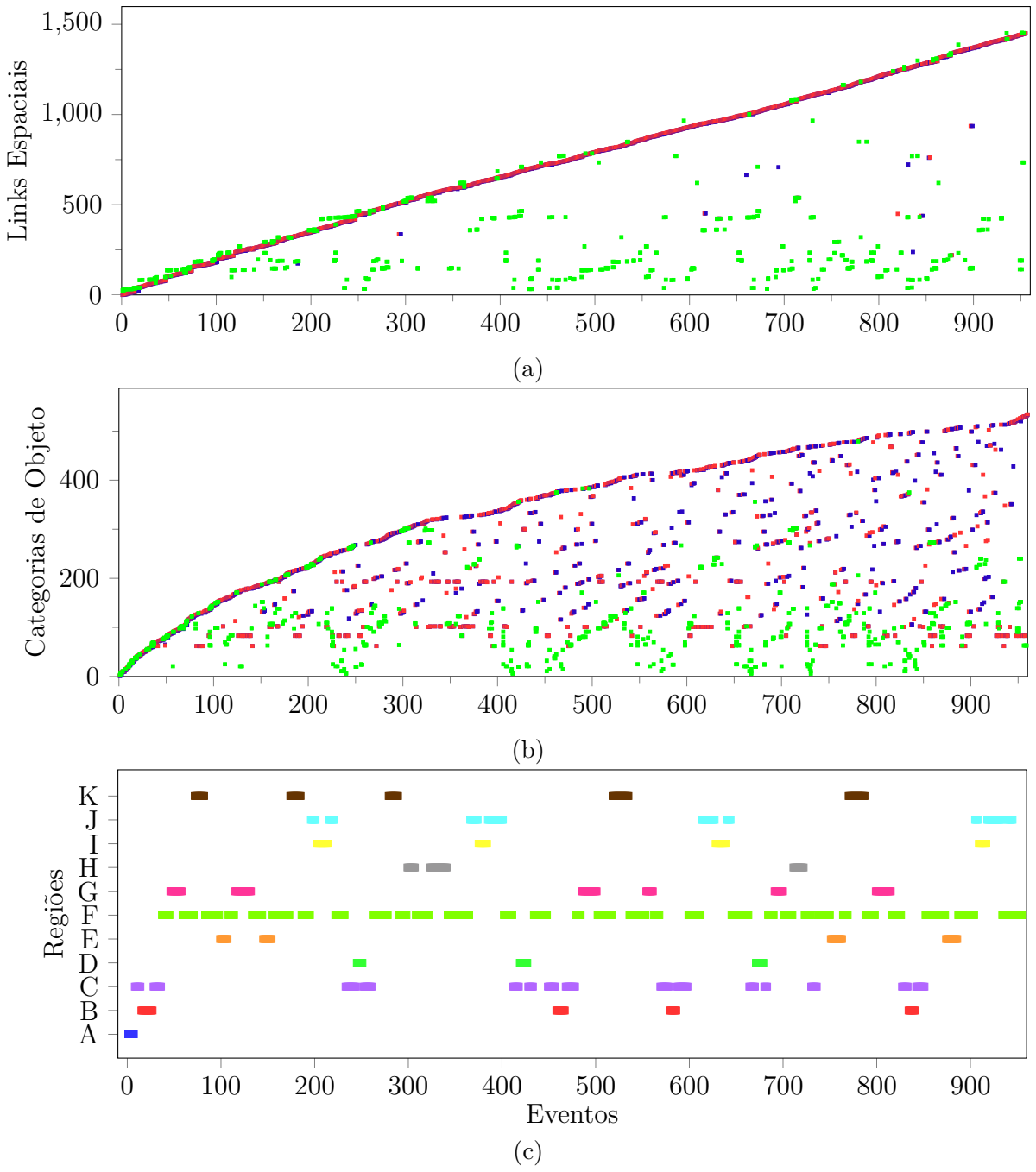


Figura 6.4 – Relação entre nós de eventos e as informações dos links espaciais conectados a eles.(a) Links espaciais conectados aos eventos na memória episódica; (b) Categorias de objetos referidos pelos links espaciais do evento; (c) Região ocupada pelo agente durante o evento. (a,b) Pontos verdes indicam uma conexão do tipo contexto espacial. Pontos azuis indicam uma conexão do tipo início ou fim de evento. Linhas tracejadas vermelhas indicam os instantes de segmentação de episódios. (c) Cada cor e letra de região refere-se as segmentações adotadas na Figura 6.3.

7 Conclusão

Este trabalho teve como objetivo principal o desenvolvimento de uma arquitetura cognitiva que incorporasse mecanismos para processamento de memórias episódicas inspirados em alguns trabalhos da literatura atual sobre o funcionamento da memória humana. Em particular, buscou-se explorar o uso do *Cognitive Systems Toolkit* e da representação de conhecimento utilizando ideias computacionais (as *Ideas*) na construção destes mecanismos, adotando suas formulações conceituais e computacionais.

De forma a alcançar esse objetivo, realizou-se um estudo das principais definições e mecanismos da memória episódica em humanos em trabalhos da neurociéncia e pscicologia. Inspirados pelos mesmos e por outros trabalhos de implementação computacional da memória episódica, optou-se pela decomposição do mecanismo de memória episódica em três etapas: codificação, responsável por identificar os episódios e criar uma representação para ele; armazenamento, responsável por adaptar e integrar o episódio criado à memória de longo prazo; e recuperação, que permite buscar por episódios nas memória a partir de informações parciais.

Assim, neste trabalho é proposto um modelo de arquitetura cognitiva baseada em *codelets* e incorporando algumas das principais teorias atuais da memória episódica humana. Destaca-se a definição de mecanismos paralelos de segmentação temporal baseados na teoria de Segmentação de Eventos e na teoria de Mudança Contextual, as quais propõem visões diferentes dos mecanismos adotados pelo cérebro humano. A forma de representação do conhecimento utilizando *Ideas* demonstrou-se flexível para criação das estruturas de informação ao longo de toda arquitetura, além de se adequar bem às teorias utilizadas, permitindo a proposta de um modelo que concilia as duas teorias de segmentação temporal utilizadas como inspiração.

Uma implementação piloto do modelo proposto foi criada para controle de um agente em ambiente simulado. Variações na sensibilidade da arquitetura para detecção de eventos e categorias de objetos mostram sua capacidade de ser adequado a aplicações para representação das variações perceptuais em eventos e episódios. Além disso, a incorporação de episódios à memória episódica mostra-se capaz de reutilizar conhecimentos já

existentes. O módulo de recuperação de episódios foi robusto na busca de episódios com bases em pistas simples e episódios completos.

Como principais contribuições deste trabalho, o modelo de arquitetura proposto carrega informações das transformações ocorridas com os objetos observados na forma de eventos em unidades de representação em alto nível. Além disso, a segmentação temporal dos eventos e episódios utiliza dois mecanismos paralelos inspirados em evidências neurobiológicas do funcionamento da memória humana. Por fim, a codificação dos episódios incorpora informações perceptuais do ambiente junto a informações internas de processos cognitivos.

7.1 Trabalhos Futuros

Uma limitação evidente do modelo proposto é o crescimento contínuo da memória episódica ao longo do tempo. Assim, a primeira melhoria a ser explorada como trabalho futuro é a incorporação de mecanismos de manutenção da memória episódica. Comumente, tais mecanismos removem episódios pouco relevantes da memória, porém a exploração de técnicas para detecção de padrões de episódio e criação de categorias de episódios permitiria a abstração de episódios pouco relevantes em uma representação genérica, semelhante ao processo de assimilação de objetos em categorias de objetos.

Derivado do problema anterior, a arquitetura piloto implementada utiliza mecanismos de aprendizado simples e de forma esparsa. Métodos de aprendizado poderiam ser aplicados nos mecanismos de identificação e detecção de regiões do ambiente, na detecção de classes de objeto, no aprendizado de categorias de eventos, na incorporação de episódios a memória de longo-prazo, entre outros. Uma pesquisa de métodos de aprendizado poderia ser realizada para identificar mecanismos interessantes a serem incorporados à arquitetura. A utilização desses mecanismos implicaria na implementação de novas arquiteturas de teste a partir do modelo base. Tais mecanismos podem também requerer um tipo diferente de representação de eventos e episódios, o que levaria a exploração de variações na representação dos eventos e episódios.

Os cenários de simulação dos experimentos foram desenvolvidos somente para validação dos módulos propostos e implementados. Novos cenários podem ser construídos para comparação do desempenho da arquitetura com memória episódica contra uma arquitetura sem memória episódica em tarefas com diferentes complexidades.

Por fim, uma das grandes dificuldades na avaliação de arquiteturas cognitivas é a comparação do desempenho de diferentes arquiteturas. Devido a flexibilidade na construção de arquiteturas cognitivas fornecida pelo CST, torna-se interessante como trabalho futuro a implementação do modelo proposto para as diferentes tarefas propostas como método de avaliação das demais arquiteturas da literatura. Com isso, o modelo proposto poderia ser comparado individualmente com outras arquiteturas, melhor explorando seus pontos fortes e fracos.

Referências

ADDIS, D. R. Are episodic memories special? On the sameness of remembered and imagined event simulation. *Journal of the Royal Society of New Zealand*, Taylor & Francis, v. 48, n. 2-3, p. 64–88, jul. 2018. ISSN 0303-6758. Citado na página 27.

ALLEN, J. F. Maintaining knowledge about temporal intervals. *Communications of the ACM*, v. 26, n. 11, p. 832–843, nov. 1983. ISSN 0001-0782, 1557-7317. Citado na página 48.

BAARS, B. J.; FRANKLIN, S. Consciousness is computational: the lida model of global workspace theory. *International Journal of Machine Consciousness*, v. 01, n. 01, p. 23–32, jun. 2009. ISSN 1793-8430. Publisher: World Scientific Publishing Co. Disponível em: <https://www.worldscientific.com/doi/abs/10.1142/S1793843009000050>. Citado na página 36.

BALDASSANO, C.; CHEN, J.; ZADBOOD, A.; PILLOW, J. W.; HASSON, U.; NORMAN, K. A. Discovering Event Structure in Continuous Narrative Perception and Memory. *Neuron*, Elsevier BV, v. 95, n. 3, p. 709–721.e5, 2017. ISSN 0896-6273. Citado 3 vezes nas páginas 7, 25 e 26.

BAUMEISTER, R. F. Toward a general theory of motivation: Problems, challenges, opportunities, and the big picture. *Motivation and Emotion*, v. 40, n. 1, p. 1–10, fev. 2016. ISSN 1573-6644. Citado 2 vezes nas páginas 39 e 40.

BERNECKER, S. (Ed.). *The Routledge handbook of philosophy of memory*. London ; New York: Routledge, Taylor & Francis Group, 2017. (Routledge handbooks in philosophy). ISBN 978-1-138-90936-6. Citado na página 22.

BONNICI, V.; GIUGNO, R.; PULVIRENTI, A.; SHASHA, D.; FERRO, A. A subgraph isomorphism algorithm and its application to biochemical data. *BMC Bioinformatics*, v. 14, n. 7, p. S13, abr. 2013. ISSN 1471-2105. Citado na página 78.

BROM, C.; PEŠKOVÁ, K.; LUKAVSKÝ, J. What Does Your Actor Remember? Towards Characters with a Full Episodic Memory. In: CAVAZZA, M.; DONIKIAN, S. (Ed.). *Virtual Storytelling. Using Virtual Reality Technologies for Storytelling*. Berlin, Heidelberg: Springer, 2007. (Lecture Notes in Computer Science), p. 89–101. ISBN 978-3-540-77039-8. Citado na página 32.

BROOKS, R. A robust layered control system for a mobile robot. *IEEE Journal on Robotics and Automation*, v. 2, n. 1, p. 14–23, mar. 1986. ISSN 2374-8710. Conference Name: IEEE Journal on Robotics and Automation. Citado na página 38.

CAMARGO, E.; SAKABE, E. Y.; GUDWIN, R. Existence, Hypotheses and Categories in Knowledge Representation. *Procedia Computer Science*, v. 213, p. 496–503, jan. 2022. ISSN 1877-0509. Disponível em: <https://www.sciencedirect.com/science/article/pii/S1877050922017872>. Citado 6 vezes nas páginas 7, 18, 40, 42, 43 e 45.

CHENG, S.; WERNING, M.; SUDDENDORF, T. Dissociating memory traces and scenario construction in mental time travel. *Neuroscience & Biobehavioral Reviews*, v. 60, p. 82–89, jan. 2016. ISSN 0149-7634. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0149763415301603>. Citado 3 vezes nas páginas 17, 27 e 28.

CLEWETT, D.; DAVACHI, L. The Ebb and Flow of Experience Determines the Temporal Structure of Memory. *Current opinion in behavioral sciences*, v. 17, p. 186–193, out. 2017. ISSN 2352-1546. Disponível em: <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5739077/>. Citado 2 vezes nas páginas 24 e 25.

CLEWETT, D.; DUBROW, S.; DAVACHI, L. Transcending time in the brain: How event memories are constructed from experience. *Hippocampus*, v. 29, n. 3, p. 162–183, mar. 2019. ISSN 1050-9631. Disponível em: <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6629464/>. Citado na página 25.

CONWAY, M. A. Memory and the self. *Journal of Memory and Language*, v. 53, n. 4, p. 594–628, out. 2005. ISSN 0749-596X. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0749596X05000987>. Citado na página 23.

CONWAY, M. A.; M. A. Conway; JUSTICE, L. V.; L. V. Justice; D'ARGEMBEAU, A. The Self-Memory System revisited: Past, present, and future. jan. 2019. MAG ID: 2989805670. Citado na página 24.

CORCHADO, F. F. R.; FRAGA, A. C. L.; SALAZAR, R. S.; CORCHADO, M. A. R.; MENDOZA, O. B. Cognitive Pervasive Service Composition Applied to Predatory Crime Deterrence. *Applied Sciences*, Multidisciplinary Digital Publishing Institute, v. 11, n. 4, p. 1803, jan. 2021. ISSN 2076-3417. Citado na página 32.

DEUTSCH, T.; GRUBER, A.; LANG, R.; VELIK, R. Episodic memory for autonomous agents. In: *2008 Conference on Human System Interactions*. [S.l.: s.n.], 2008. p. 621–626. ISSN 2158-2254. Citado na página 32.

DINGS, R.; NEWEN, A. Constructing the Past: the Relevance of the Narrative Self in Modulating Episodic Memory. *Review of Philosophy and Psychology*, ago. 2021. ISSN 1878-5166. Disponível em: <https://doi.org/10.1007/s13164-021-00581-2>. Citado na página 27.

DODD, W.; GUTIERREZ, R. The role of episodic memory and emotion in a cognitive robot. In: *ROMAN 2005. IEEE International Workshop on Robot and Human Interactive Communication, 2005.* [S.l.: s.n.], 2005. p. 692–697. ISSN 1944-9437. Citado na página 30.

EZZYAT, Y.; DAVACHI, L. What Constitutes an Episode in Episodic Memory? *Psychological science*, v. 22, n. 2, p. 243–252, fev. 2011. ISSN 0956-7976. Disponível em: <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4451827/>. Citado na página 23.

FRANKLIN, S.; GRAESSER, A. Is It an agent, or just a program?: A taxonomy for autonomous agents. In: MÜLLER, J. P.; WOOLDRIDGE, M. J.; JENNINGS, N. R. (Ed.). *Intelligent Agents III Agent Theories, Architectures, and Languages*. Berlin, Heidelberg: Springer, 1997. p. 21–35. ISBN 978-3-540-68057-4. Citado na página 28.

FRANKLIN, S.; MADL, T.; STRAIN, S.; FAGHIHI, U.; DONG, D.; KUGELE, S.; SNAIDER, J.; AGRAWAL, P.; CHEN, S. A LIDA cognitive model tutorial. *Biologically Inspired Cognitive Architectures*, v. 16, p. 105–130, abr. 2016. ISSN 2212-683X. Citado na página 34.

GILBOA, A.; MARLATTE, H. Neurobiology of schemas and schema-mediated memory. *Trends in Cognitive Sciences*, v. 21, n. 8, p. 618–631, Aug 2017. ISSN 1364-6613. Citado na página 17.

GREGORIĆ, J.; SEDER, M.; PETROVIĆ, I. Autonomous Hierarchy Creation for Path Planning of Mobile Robots in Large Environments. In: PETROVIC, I.; MENEGATTI, E.; MARKOVIĆ, I. (Ed.). *Intelligent Autonomous Systems 17*. Cham: Springer Nature Switzerland, 2023. p. 909–922. ISBN 978-3-031-22216-0. Citado na página 64.

GUDWIN, R.; PARAENSE, A.; de Paula, S. M.; FRÓES, E.; GIBAUT, W.; CASTRO, E.; FIGUEIREDO, V.; RAIZER, K. The Multipurpose Enhanced Cognitive Architecture (MECA). *Biologically Inspired Cognitive Architectures*, v. 22, p. 20–34, out. 2017. ISSN 2212-683X. Citado 5 vezes nas páginas 7, 34, 37, 38 e 39.

GUDWIN, R. R. A Review of Motivational Systems and Emotions in Cognitive Architectures and Systems. In: OSIPOV, G. S.; PANOV, A. I.; YAKOVLEV, K. S. (Ed.). *Artificial Intelligence: 5th RAAI Summer School, Dolgoprudny, Russia, July 4–7, 2019, Tutorial Lectures*. Cham: Springer International Publishing, 2019, (Lecture Notes in Computer Science). p. 65–84. ISBN 978-3-030-33274-7. Disponível em: <https://doi.org/10.1007/978-3-030-33274-7_4>. Citado 2 vezes nas páginas 39 e 40.

GüLER, B.; ADıGüZEL, Z.; UYSAL, B.; GUNSELI, E. *Discrete memories of a continuous world: A working memory perspective on event segmentation*. PsyArXiv, 2023. Disponível em: <https://psyarxiv.com/4mx79/>. Citado na página 23.

HAIKONEN, P. O.; HAIKONEN, P. O. A. *Consciousness and Robot Sentience*. Singapore: World Scientific, 2012. (Series on Machine Consciousness, 2). ISBN 978-981-4407-15-1. Citado na página 34.

HULL, C. L. *Principles of Behavior: An Introduction to Behavior Theory*. Oxford, England: Appleton-Century, 1943. x, 422 p. (Principles of Behavior: An Introduction to Behavior Theory.). Citado na página 39.

KAHANA, M. J. Computational Models of Memory Search. *Annual Review of Psychology*, v. 71, n. 1, p. 107–138, 2020. _eprint: https://doi.org/10.1146/annurev-psych-010418-103358. Disponível em: <https://doi.org/10.1146/annurev-psych-010418-103358>. Citado 2 vezes nas páginas 24 e 27.

KOTSERUBA, I.; TSOTSOS, J. K. 40 years of cognitive architectures: Core cognitive abilities and practical applications. *Artificial Intelligence Review*, v. 53, n. 1, p. 17–94, jan. 2020. ISSN 1573-7462. Citado 3 vezes nas páginas 17, 29 e 30.

KUPPUSWAMY, N. S.; CHO, S.-h.; KIM, J.-h. A Cognitive Control Architecture for an Artificial Creature using Episodic Memory. In: *2006 SICE-ICASE International Joint Conference*. [S.l.: s.n.], 2006. p. 3104–3110. Citado na página 30.

LAIRD, J. E. *The Soar cognitive architecture*. Cambridge, Mass.: MIT Press, 2012. (A bradford book). ISBN 978-0-262-12296-2. Citado 3 vezes nas páginas 7, 28 e 30.

LAIRD, J. E.; NEWELL, A.; ROSENBLOOM, P. S. SOAR: An architecture for general intelligence. *Artificial Intelligence*, v. 33, n. 1, p. 1–64, set. 1987. ISSN 0004-3702. Disponível em: <https://www.sciencedirect.com/science/article/pii/0004370287900506>. Citado na página 29.

LANGLEY, P.; THOMPSON, K.; IBA, W.; GENNARI, J. H.; ALLEN, J. A. *An Integrated Cognitive Architecture for Autonomous Agents*. Fort Belvoir, VA, 1989. Disponível em: <http://www.dtic.mil/docs/citations/ADA225701>. Citado na página 31.

LOCKE, J. *An essay concerning human understanding*. [S.l.]: Kay & Troutman, 1847. Citado na página 40.

MAC, T. T.; COPOT, C.; TRAN, D. T.; KEYSER, R. D. A hierarchical global path planning approach for mobile robots based on multi-objective particle swarm optimization. *Applied Soft Computing*, v. 59, p. 68–76, out. 2017. ISSN 1568-4946. Citado na página 64.

MAHR, J. B. The dimensions of episodic simulation. *Cognition*, v. 196, p. 104085, mar. 2020. ISSN 0010-0277. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0010027719302598>. Citado na página 23.

MAHR, J. B.; CSIBRA, G. Why do we remember? The communicative function of episodic memory. *The Behavioral and brain sciences*, v. 41, p. e1, 2018. ISSN 0140-525X. Disponível em: <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5404722/>. Citado 3 vezes nas páginas 17, 18 e 23.

MARTIN, L.; JAIME, K.; RAMOS, F.; ROBLES, F. Bio-inspired cognitive architecture of episodic memory. *Cognitive Systems Research*, v. 76, p. 26–45, dez. 2022. ISSN 1389-0417. Disponível em: <https://www.sciencedirect.com/science/article/pii/S1389041722000390>. Citado 3 vezes nas páginas 17, 32 e 33.

MARTIN, L.; ROSALES, J. H.; JAIME, K.; RAMOS, F. Affective episodic memory system for virtual creatures: The first step of emotion-oriented memory. *Computational Intelligence and Neuroscience*, v. 2021, 2021. Citado 2 vezes nas páginas 7 e 33.

MICHAELIAN, K. *Mental Time Travel: Episodic Memory and Our Knowledge of the Personal Past*. [S.l.]: The MIT Press, 2016. v. 1. Citado 2 vezes nas páginas 24 e 27.

MICHAELIAN, K.; SUTTON, J. Memory. In: ZALTA, E. N. (Ed.). *The Stanford Encyclopedia of Philosophy*. Summer 2017. Metaphysics Research Lab, Stanford University, 2017. Disponível em: <https://plato.stanford.edu/archives/sum2017/entries/memory/>. Citado 2 vezes nas páginas 23 e 28.

MéNAGER, D. H.; CHOI, D.; ROBINS, S. K. A Hybrid Theory of Event Memory. *Minds and Machines*, v. 32, n. 2, p. 365–394, jun. 2022. ISSN 1572-8641. Disponível em: <https://doi.org/10.1007/s11023-021-09578-3>. Citado 4 vezes nas páginas 7, 17, 31 e 32.

NEWELL, A.; SIMON, H. GPS, A Program that Simulates Human Thought. In: *Readings in Cognitive Science*. Elsevier, 1961. p. 453–460. ISBN 978-1-4832-1446-7. Disponível em: <https://linkinghub.elsevier.com/retrieve/pii/B9781483214467500406>. Citado na página 28.

NUXOLL, A. M.; LAIRD, J. E. Enhancing intelligent agents with episodic memory. *Cognitive Systems Research*, v. 17-18, p. 34–48, jul. 2012. ISSN 13890417. Disponível em: <https://linkinghub.elsevier.com/retrieve/pii/S1389041711000428>. Citado na página 17.

PARAENSE, A. L. O.; RAIZER, K.; PAULA, S. M. de; ROHMER, E.; GUDWIN, R. R. The cognitive systems toolkit and the CST reference cognitive architecture. *Biologically Inspired Cognitive Architectures*, v. 17, p. 32–48, jul. 2016. ISSN 2212-683X. Disponível em: <https://www.sciencedirect.com/science/article/pii/S2212683X1630038X>. Citado 4 vezes nas páginas 7, 18, 36 e 37.

PARK, G.-M.; YOO, Y.-H.; KIM, D.-H.; KIM, J.-H. Deep ART Neural Model for Biologically Inspired Episodic Memory and Its Application to Task Performance of Robots. *IEEE Transactions on Cybernetics*, v. 48, n. 6, p. 1786–1799, jun. 2018. ISSN 2168-2275. Conference Name: IEEE Transactions on Cybernetics. Citado na página 17.

PROCOPIUC, O.; AGARWAL, P. K.; ARGE, L.; VITTER, J. S. Bkd-Tree: A Dynamic Scalable kd-Tree. In: HADZILACOS, T.; MANOLOPOULOS, Y.; RODDICK, J.; THEODORIDIS, Y. (Ed.). *Advances in Spatial and Temporal Databases*. Berlin, Heidelberg: Springer, 2003. p. 46–65. ISBN 978-3-540-45072-6. Citado na página 76.

ROHMER, E.; SINGH, S. P. N.; FREESE, M. Coppeliasim (formerly v-rep): a versatile and scalable robot simulation framework. In: *Proc. of The International Conference on Intelligent Robots and Systems (IROS)*. [s.n.], 2013. Disponível em: <www.coppeliarobotics.com>. Citado na página 52.

RYU, H. Hierarchical Path-Planning for Mobile Robots Using a Skeletonization-Informed Rapidly Exploring Random Tree*. *Applied Sciences*, Multidisciplinary Digital Publishing Institute, v. 10, n. 21, p. 7846, jan. 2020. ISSN 2076-3417. Citado 2 vezes nas páginas 8 e 64.

SIMON, H.; NEWELL, A. Human problem solving: The state of the theory in 1970. v. 26, p. 145–159, 1971. Disponível em: <https://iiif.library.cmu.edu/file/Newell_box00018_fld01306_doc0001/Newell_box00018_fld01306_doc0001.pdf>. Citado na página 28.

SQUIRE, L. R.; ZOLA-MORGAN, S. Memory: brain systems and behavior. *Trends in Neurosciences*, v. 11, n. 4, p. 170–175, jan. 1988. ISSN 0166-2236. Disponível em: <https://www.sciencedirect.com/science/article/pii/0166223688901440>. Citado 3 vezes nas páginas 7, 21 e 22.

SUDDENDORF, T.; CORBALLIS, M. C. Mental time travel and the evolution of the human mind. *Genetic, Social, and General Psychology Monographs*, v. 123, n. 2, p. 133–167, maio 1997. ISSN 8756-7547. Citado na página 23.

SUN, R. *A Tutorial on CLARION 5.0*. 2003. Citado na página 34.

SUN, R. The CLARION Cognitive Architecture: Extending Cognitive Modeling to Social Simulation. In: SUN, R. (Ed.). *Cognition and Multi-Agent Interaction: From Cognitive Modeling to Social Simulation*. Cambridge: Cambridge University Press, 2005. p. 79–100. ISBN 978-0-521-83964-8. Disponível em: <https://www.cambridge.org/core/books/cognition-and-multiagent-interaction/

clarion-cognitive-architecture-extending-cognitive-modeling-to-social-simulation/0873DF19A72639841BF5D9B5DEE64453>. Citado na página 36.

TECUCI, D.; PORTER, B. An Episodic Based Approach to Complex Event Processing. In: *AAAI Spring Symposium: Intelligent Event Processing*. [S.l.: s.n.], 2009. Citado na página 30.

TSIEN, J. Z. Chapter 4.1 Neural coding of episodic memory. In: DERE, E.; EASTON, A.; NADEL, L.; HUSTON, J. P. (Ed.). *Handbook of Behavioral Neuroscience*. Elsevier, 2008, (Handbook of Episodic Memory, v. 18). p. 399–625. Disponível em: <https://www.sciencedirect.com/science/article/pii/S1569733908002221>. Citado na página 17.

TULVING, E. Episodic and semantic memory. In: *Organization of memory*. Oxford, England: Academic Press, 1972. p. xiii, 423–xiii, 423. Citado 7 vezes nas páginas 17, 21, 22, 23, 27, 29 e 51.

TULVING, E. Memory and consciousness. *Canadian Psychology / Psychologie canadienne*, Canadian Psychological Association, Canada, v. 26, p. 1–12, 1985. ISSN 1878-7304. Citado na página 22.

ULLMANN, J. R. An Algorithm for Subgraph Isomorphism. *Journal of the ACM*, v. 23, n. 1, p. 31–42, jan. 1976. ISSN 0004-5411. Citado na página 78.

WERNING, M. Predicting the Past from Minimal Traces: Episodic Memory and its Distinction from Imagination and Preservation. *Review of Philosophy and Psychology*, v. 11, n. 2, p. 301–333, jun. 2020. MAG ID: 3020386461. Citado 2 vezes nas páginas 23 e 27.

WHEELER, M. E.; PETERSEN, S. E.; BUCKNER, R. L. Memory's echo: Vivid remembering reactivates sensory-specific cortex. *Proceedings of the National Academy of Sciences*, v. 97, n. 20, p. 11125–11129, set. 2000. Publisher: Proceedings of the National Academy of Sciences. Disponível em: <https://www.pnas.org/doi/full/10.1073/pnas.97.20.11125>. Citado na página 27.

WILSOM, N. *Setting Up & Using the NACS*. 2013. Citado na página 34.

ZACKS, J. M. Event perception and memory. *Annual review of psychology*, v. 71, p. 165–191, jan. 2020. ISSN 0066-4308. Disponível em: <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8679009/>. Citado 2 vezes nas páginas 18 e 24.

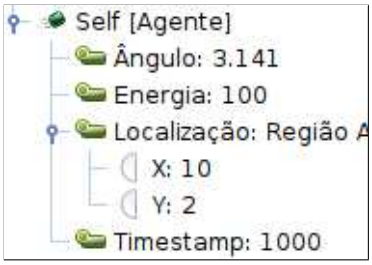
ZHENG, J.; SCHJETNAN, A. G. P.; YEBRA, M.; GOMES, B. A.; MOSHER, C. P.; KALIA, S. K.; VALIANTE, T. A.; MAMELAK, A. N.; KREIMAN, G.; RUTISHAUSER, U. Neurons detect cognitive boundaries to structure episodic memories in humans. *Nature Neuroscience*, Nature Publishing Group, v. 25, n. 3, p. 358–368, mar. 2022. ISSN 1546-1726. Citado 3 vezes nas páginas 25, 26 e 47.

ZOU, Q.; CONG, M.; LIU, D.; DU, Y. A neurobiologically inspired mapping and navigating framework for mobile robots. *Neurocomputing*, v. 460, p. 181–194, out. 2021. ISSN 0925-2312. Citado na página 33.

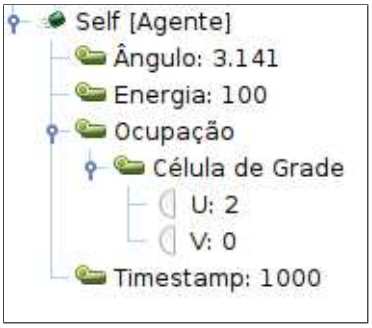
ZUO, L.; GUO, Q.; XU, X.; FU, H. A hierarchical path planning approach based on A∗ and least-squares policy iteration for mobile robots. *Neurocomputing*, v. 170, p. 257–266, dez. 2015. ISSN 0925-2312. Citado na página 64.

Apêndice A - Exemplos de *Ideas*

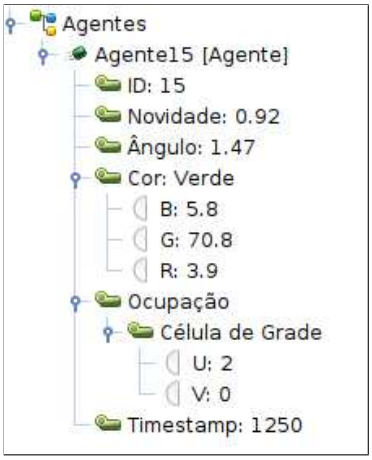
A seguir são apresentados exemplos de *Ideas* utilizadas para representação da informação nos diferentes *memory objects* e *memory containers* presentes na arquitetura implementada no Capítulo 5.



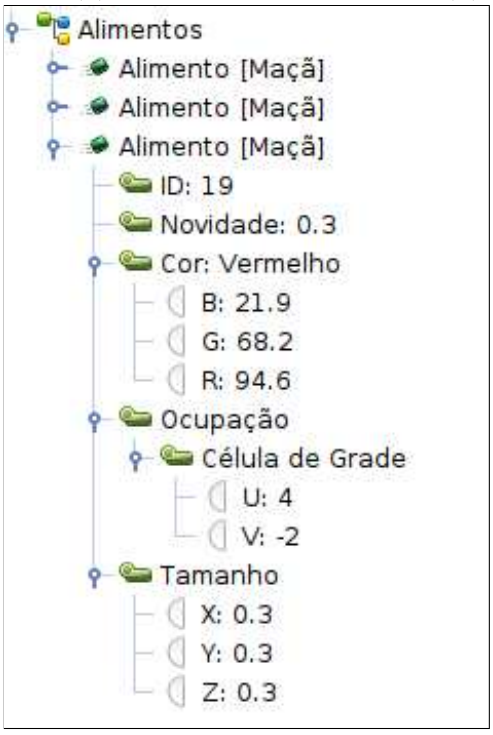
(a) Propriocepção



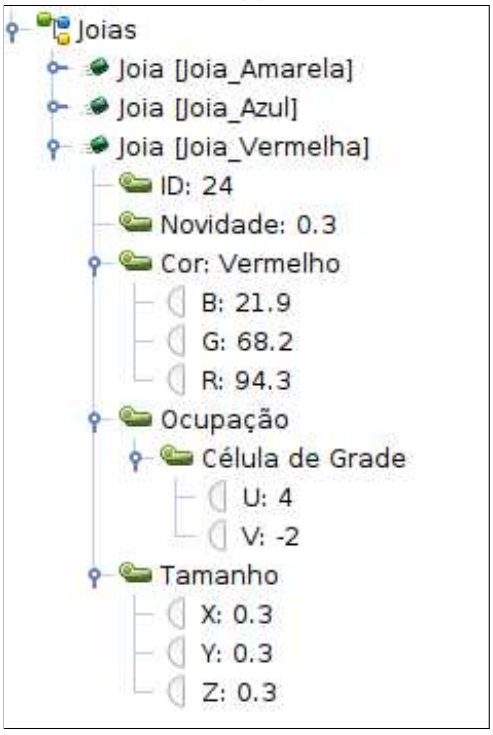
(b) Sensoriamento interno



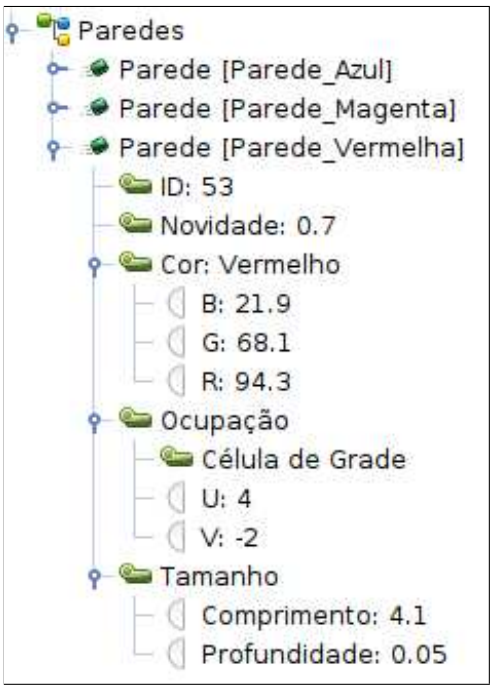
(c) Agentes



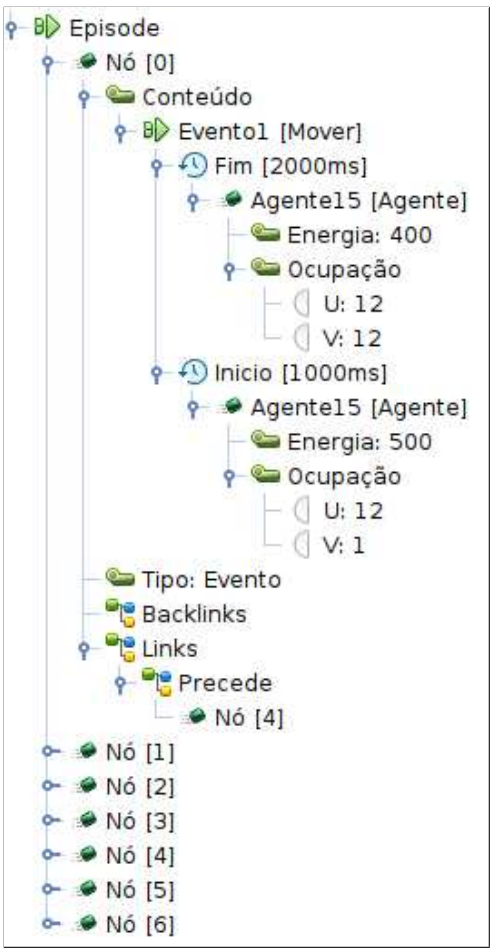
(d) Alimentos



(e) Joias

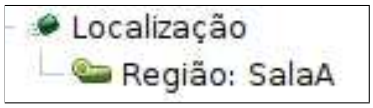


(f) Paredes



(g) Episodio

(h) Impulso



(i) Região

Figura 7.1 – Exemplos de *ideas* presentes nos *memory objects* da arquitetura implementada na Figura 5.3.

Apêndice B - Resultados detalhados dos Experimentos

B.1 Experimento 1

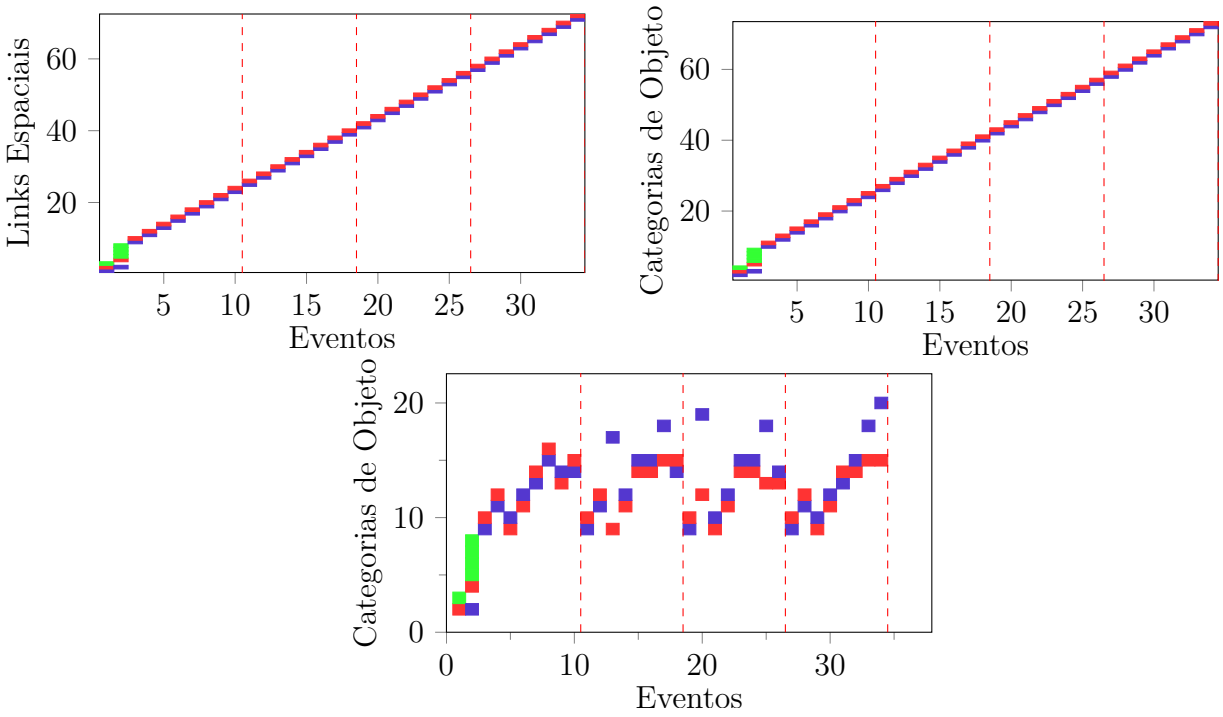


Figura 7.2 – Relações entre eventos e as informações dos *links* espaciais conectados para execução com limiar de evento 0.01 e objeto 1.0.

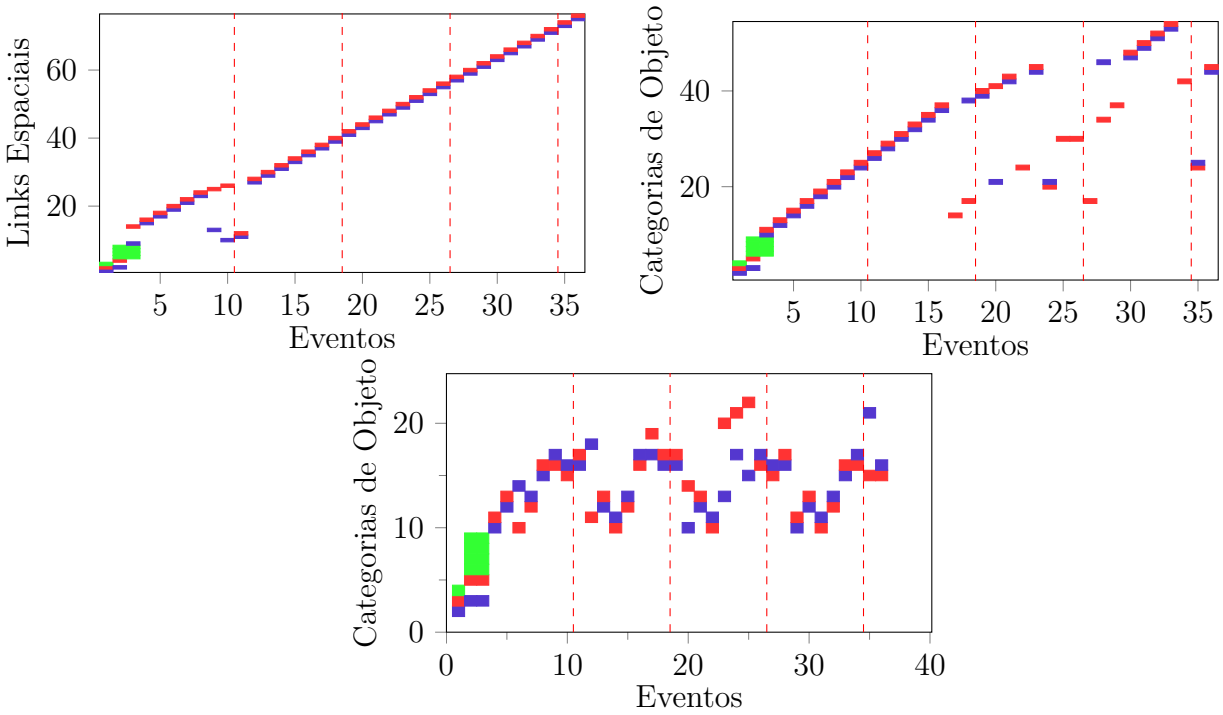


Figura 7.3 – Relações entre eventos e as informações dos *links* espaciais conectados para execução com limiar de evento 0.01 e objeto 0.99.

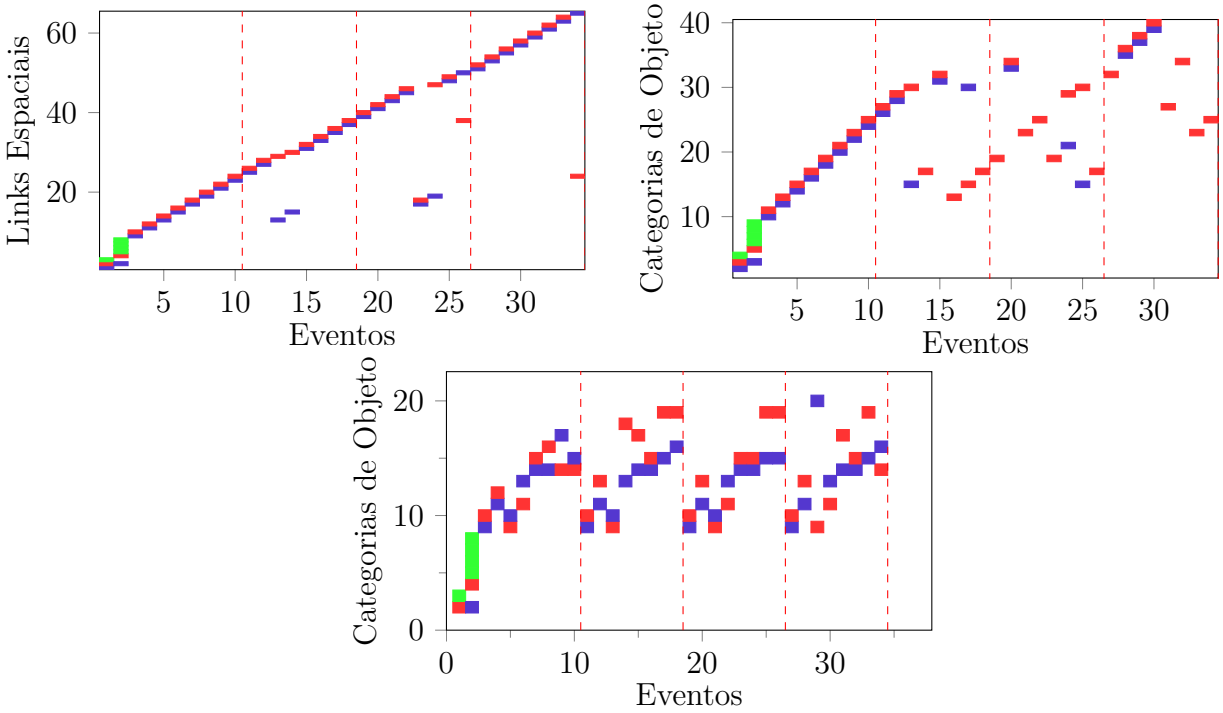


Figura 7.4 – Relações entre eventos e as informações dos *links* espaciais conectados para execução com limiar de evento 0.01 e objeto 0.95.

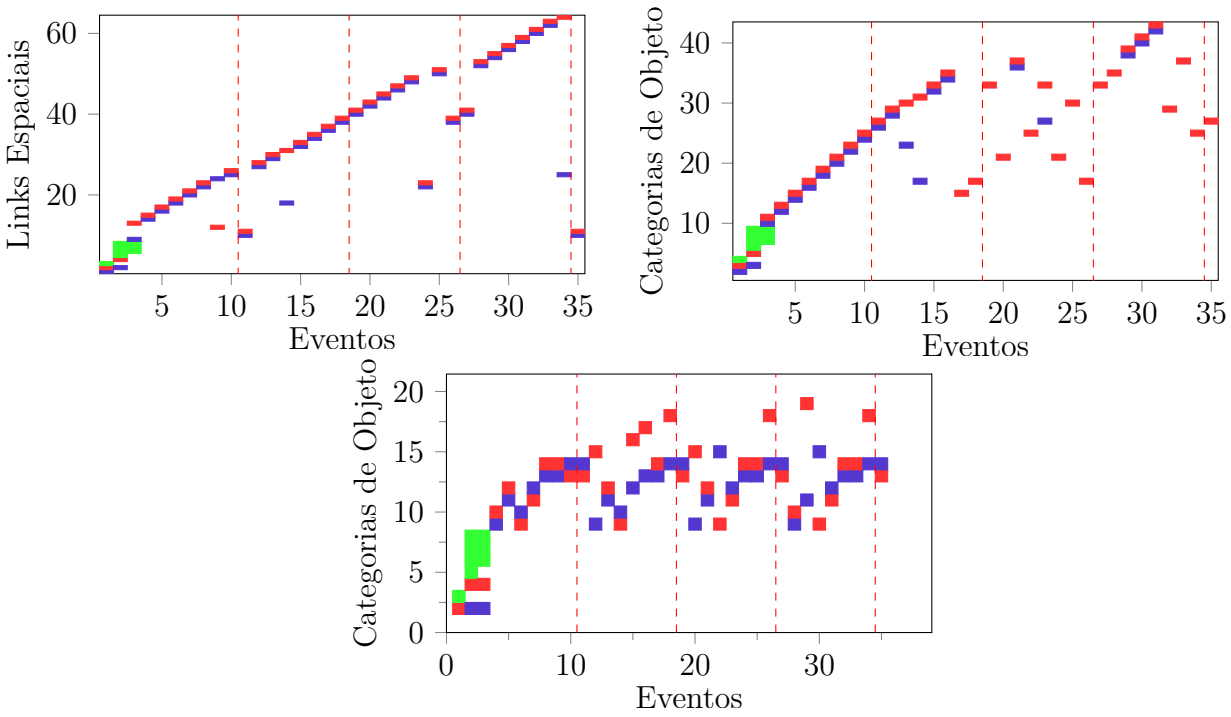


Figura 7.5 – Relações entre eventos e as informações dos *links* espaciais conectados para execução com limiar de evento 0.01 e objeto 0.9.

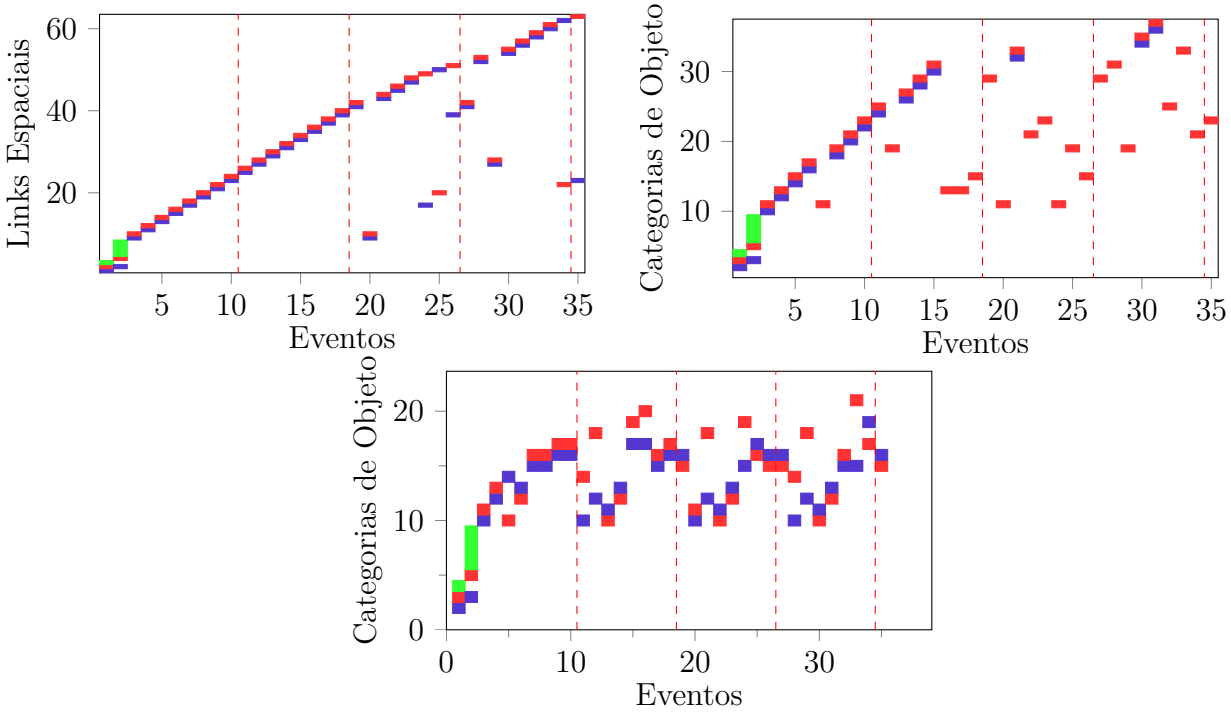


Figura 7.6 – Relações entre eventos e as informações dos *links* espaciais conectados para execução com limiar de evento 0.01 e objeto 0.8.

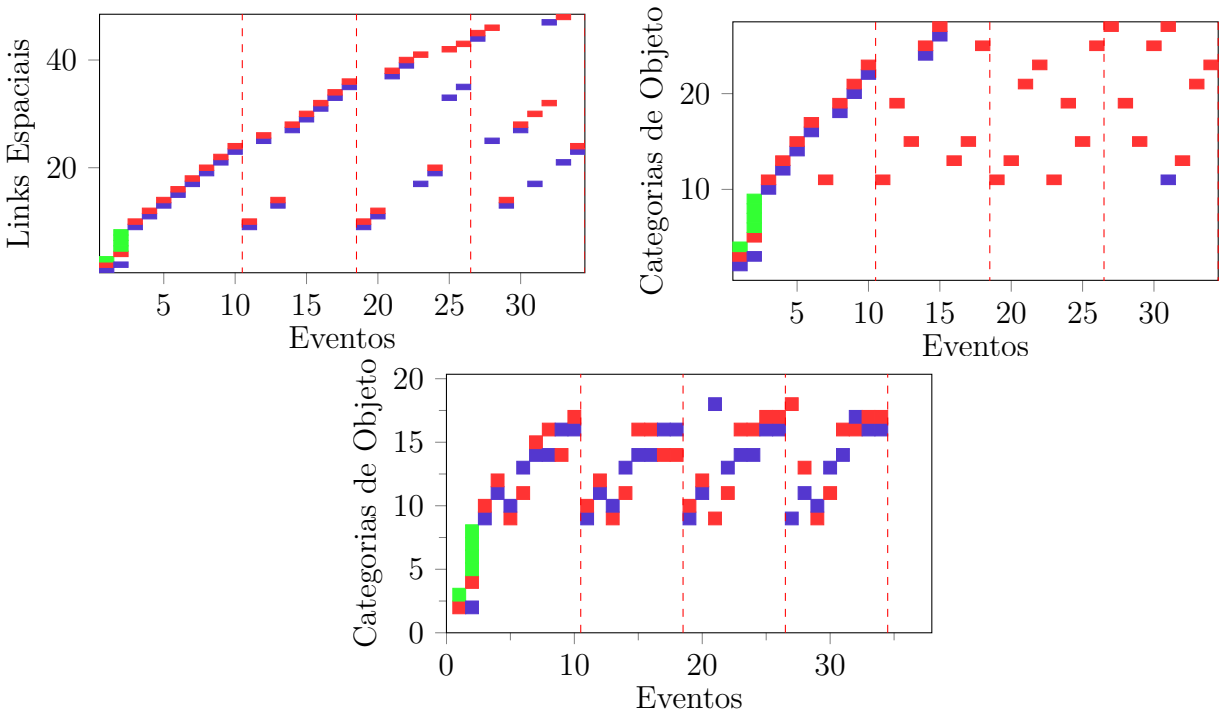


Figura 7.7 – Relações entre eventos e as informações dos *links* espaciais conectados para execução com limiar de evento 0.01 e objeto 0.7.

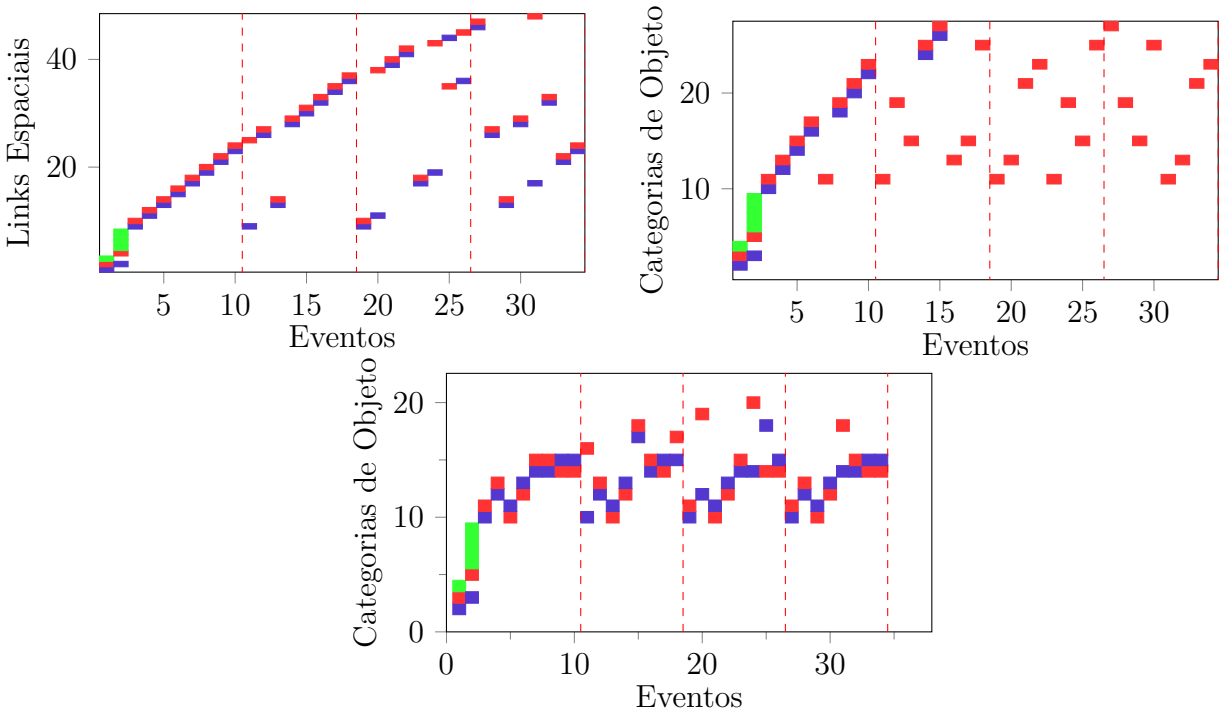


Figura 7.8 – Relações entre eventos e as informações dos *links* espaciais conectados para execução com limiar de evento 0.01 e objeto 0.6.

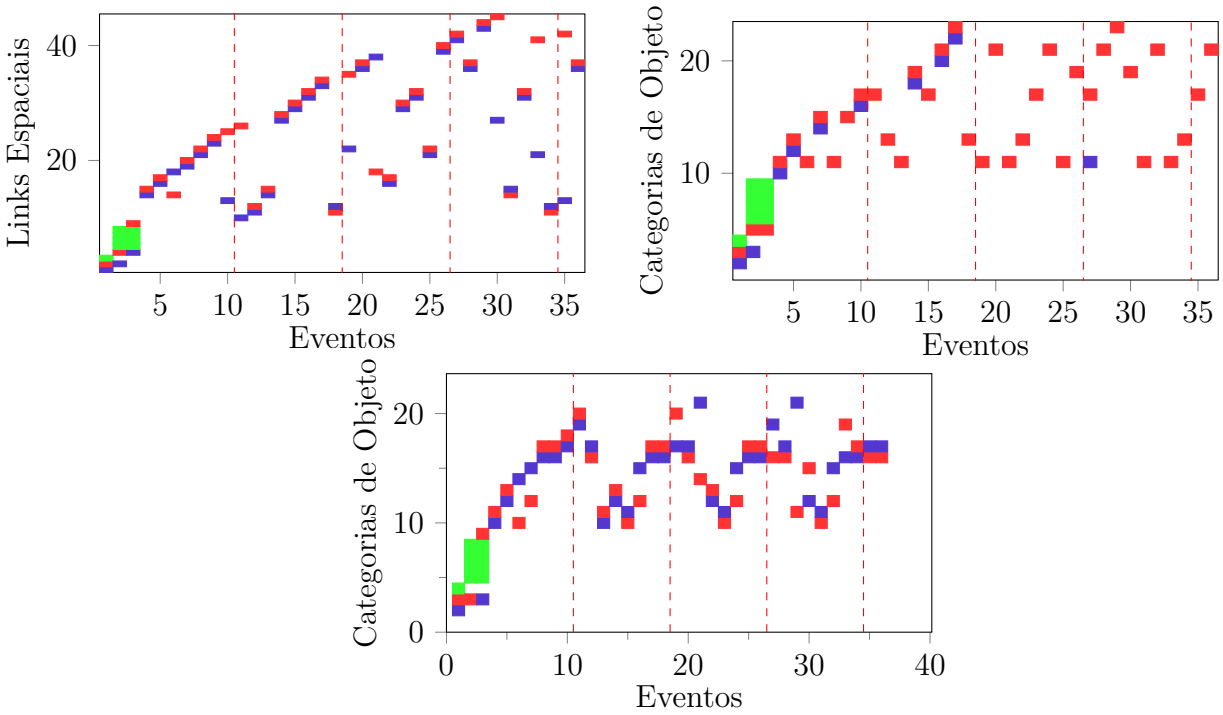


Figura 7.9 – Relações entre eventos e as informações dos *links* espaciais conectados para execução com limiar de evento 0.01 e objeto 0.5.

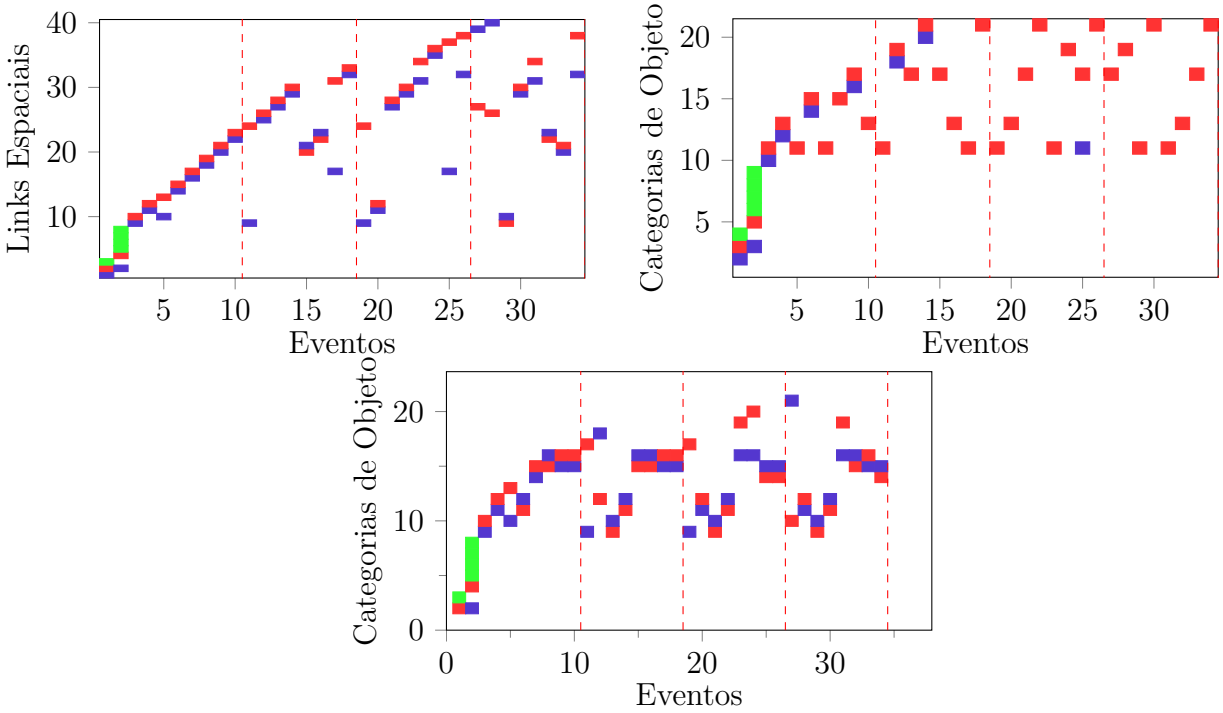


Figura 7.10 – Relações entre eventos e as informações dos *links* espaciais conectados para execução com limiar de evento 0.01 e objeto 0.4.

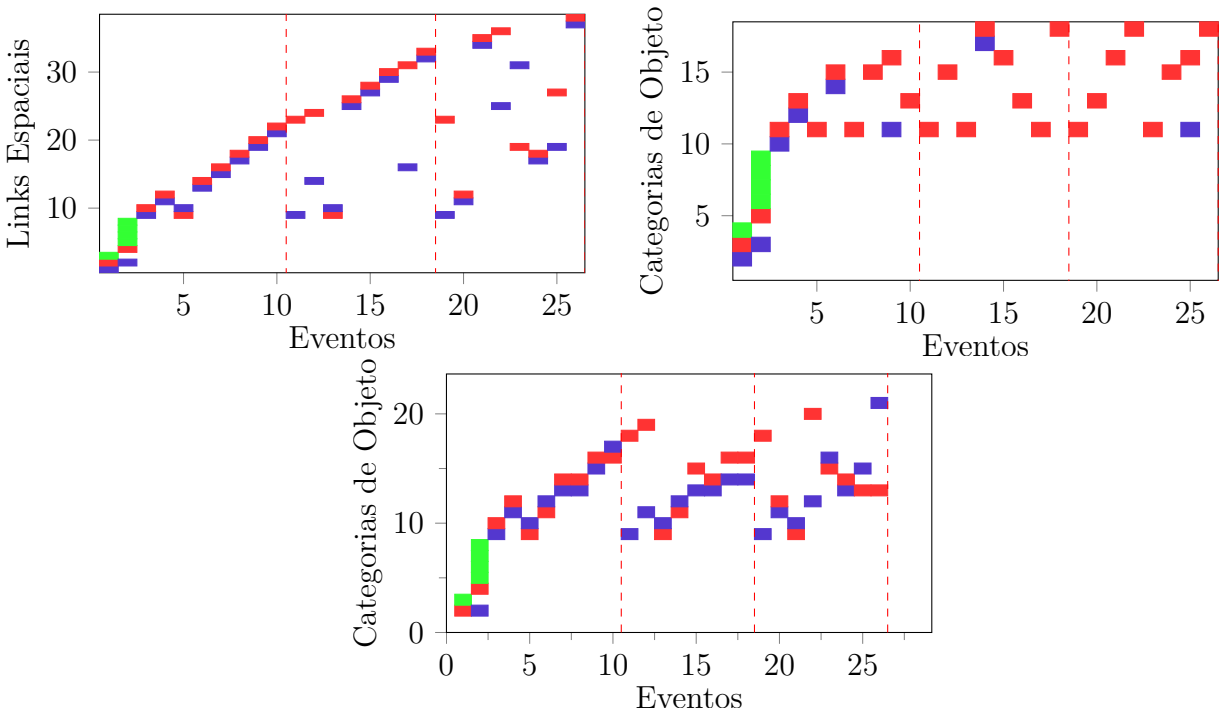


Figura 7.11 – Relações entre eventos e as informações dos *links* espaciais conectados para execução com limiar de evento 0.01 e objeto 0.3.

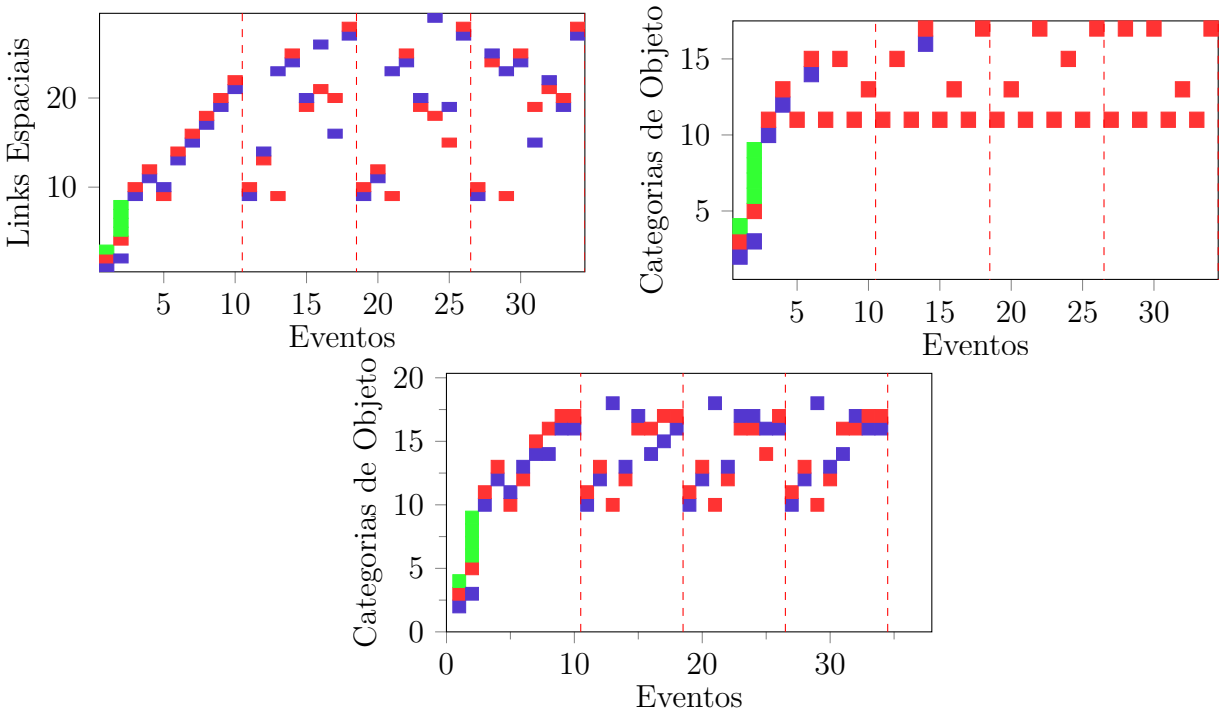


Figura 7.12 – Relações entre eventos e as informações dos *links* espaciais conectados para execução com limiar de evento 0.01 e objeto 0.2.

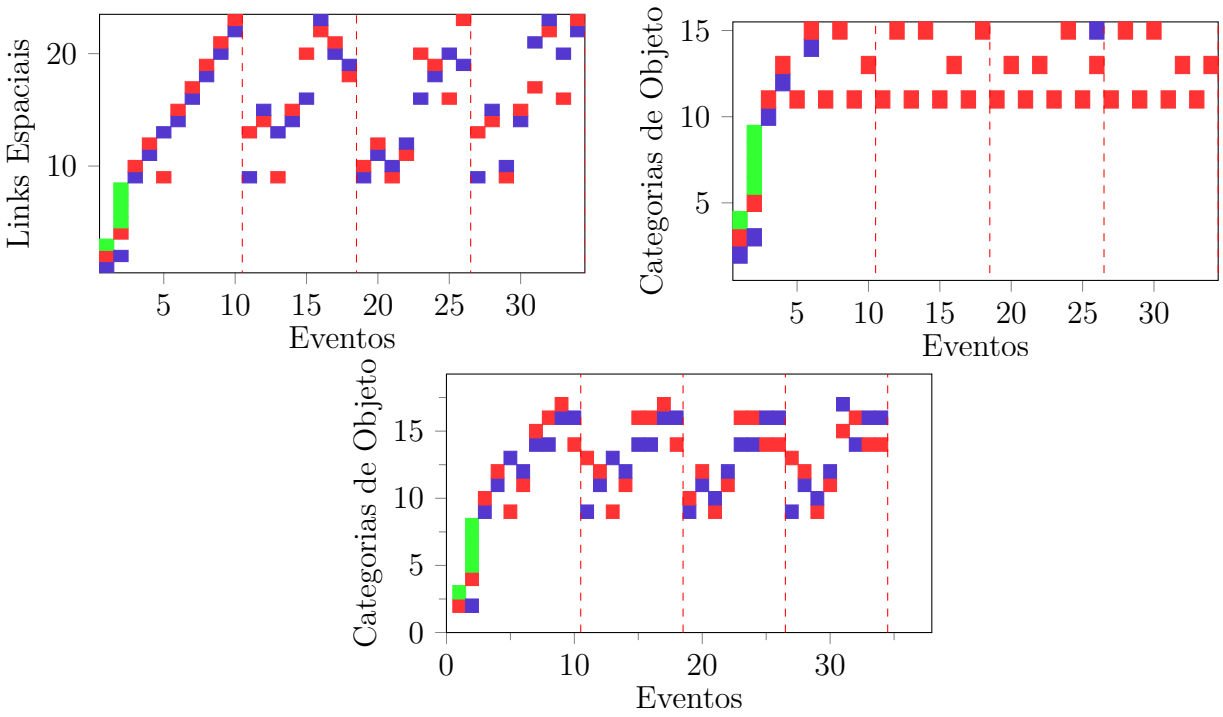


Figura 7.13 – Relações entre eventos e as informações dos *links* espaciais conectados para execução com limiar de evento 0.01 e objeto 0.1.

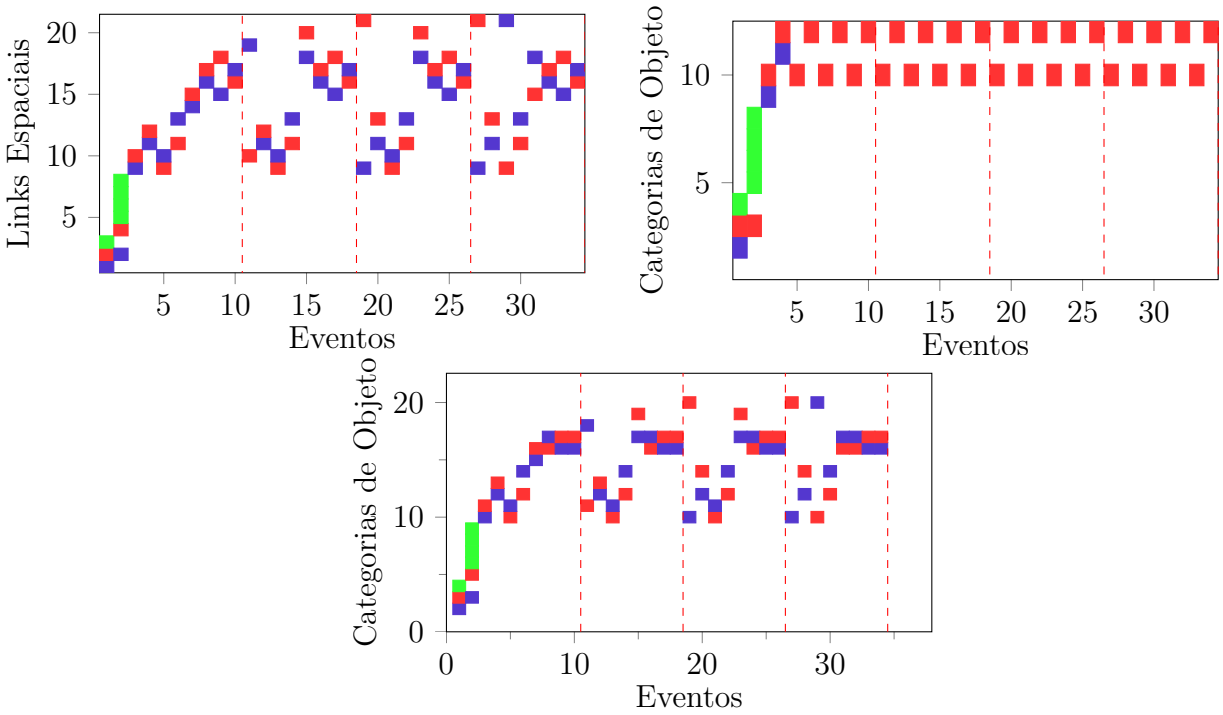


Figura 7.14 – Relações entre eventos e as informações dos *links* espaciais conectados para execução com limiar de evento 0.01 e objeto 0.0.

B.2 Experimento 3

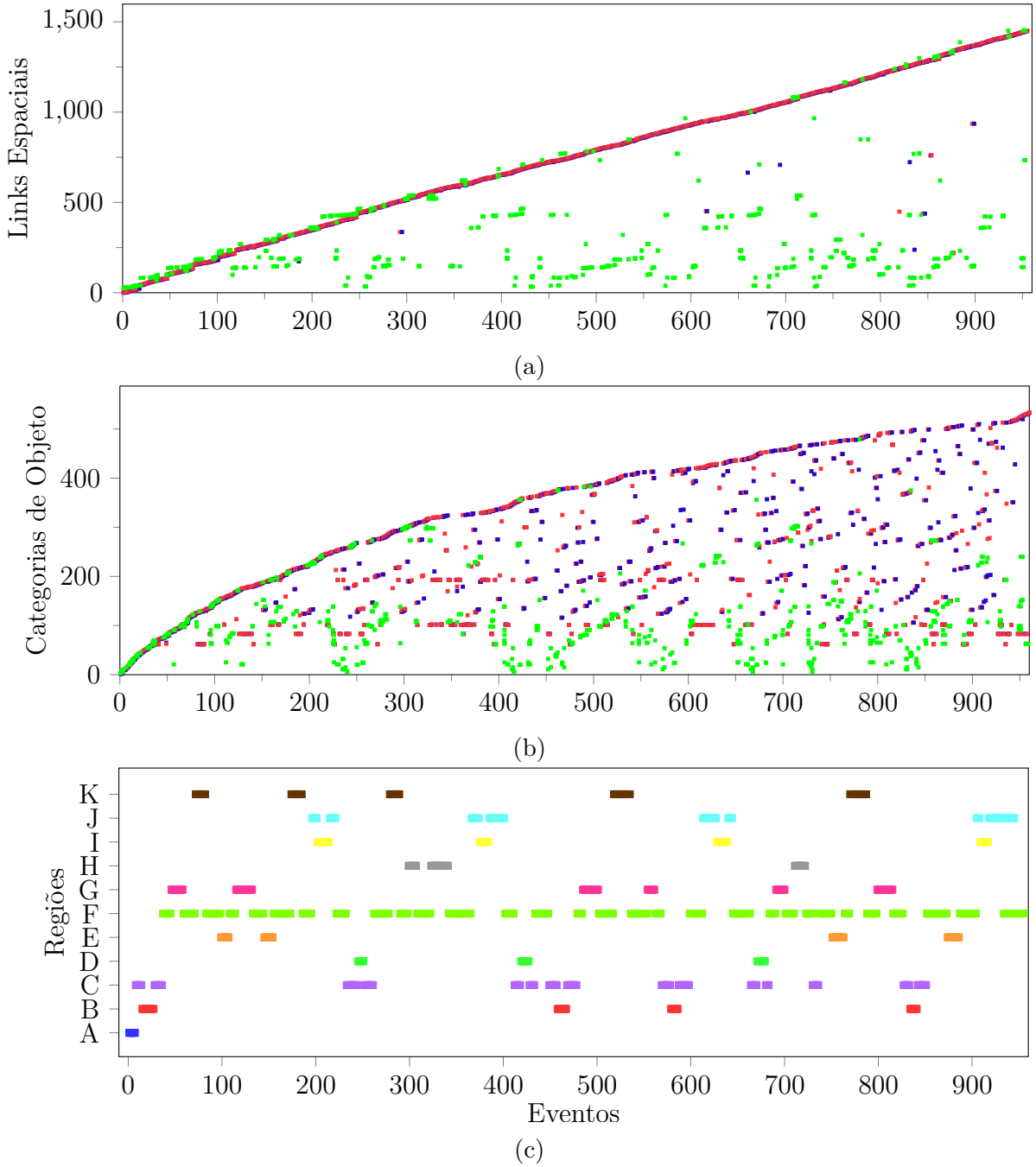


Figura 7.15 – Relação entre nós de eventos e as informações dos links espaciais conectados a eles.(a) Links espaciais conectados aos eventos na memória episódica; (b) Categorias de objetos referidos pelos links espaciais do evento; (c) Região ocupada pelo agente durante o evento. (a,b) Verdes indicam uma conexão do tipo contexto espacial. Azuis indicam uma conexão do tipo início de evento. Vermelhos indicam uma conexão do tipo final de evento. (c) Cada cor e letra de região refere-se as segmentações adotadas na Figura 6.3.

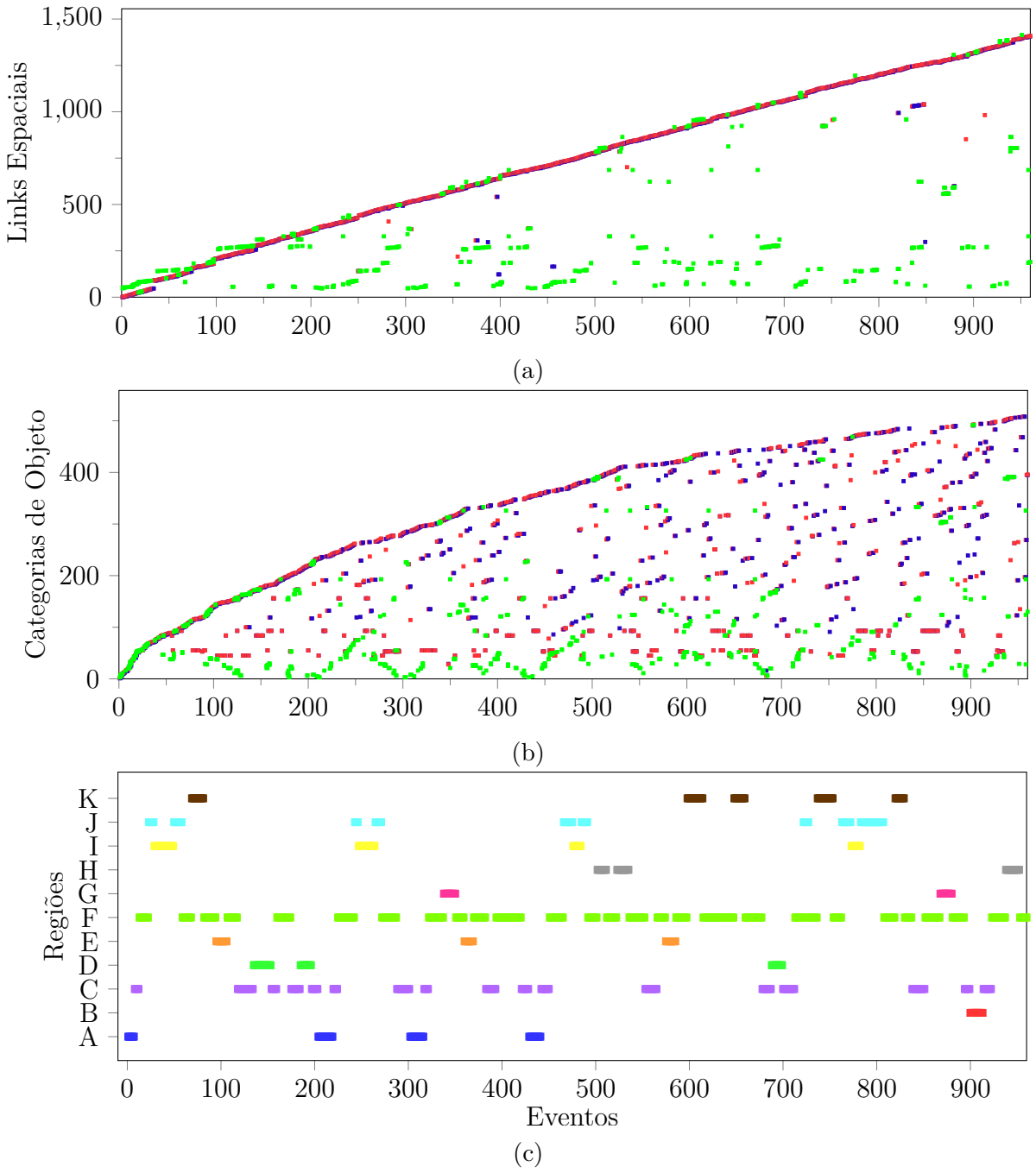


Figura 7.16 – Relação entre nós de eventos e as informações dos links espaciais conectados a eles.(a) Links espaciais conectados aos eventos na memória episódica; (b) Categorias de objetos referidos pelos links espaciais do evento; (c) Região ocupada pelo agente durante o evento. (a,b) Verdes indicam uma conexão do tipo contexto espacial. Azuis indicam uma conexão do tipo início de evento. Vermelhos indicam uma conexão do tipo final de evento. (c) Cada cor e letra de região refere-se as segmentações adotadas na Figura 6.3.

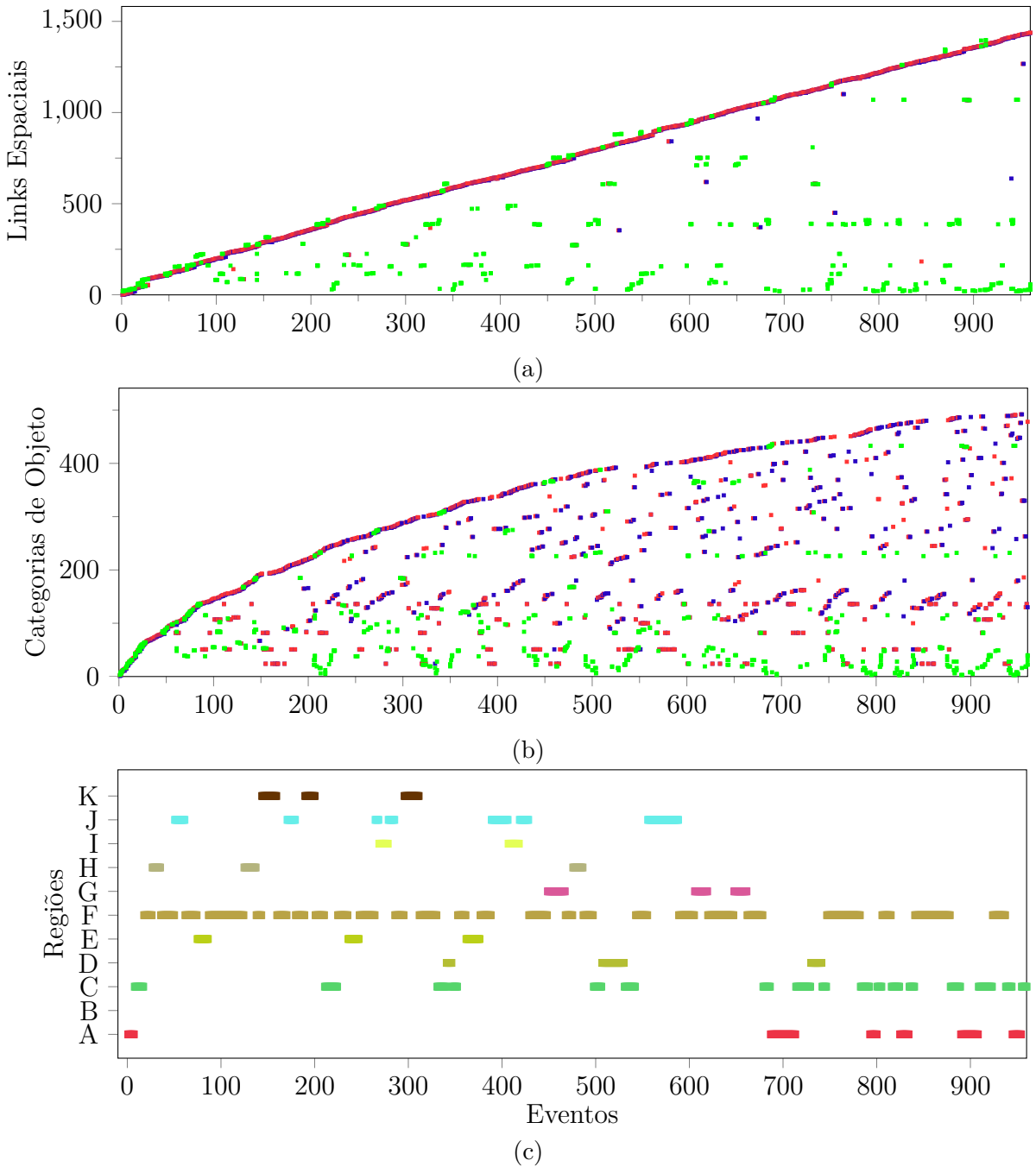


Figura 7.17 – Relação entre nós de eventos e as informações dos links espaciais conectados a eles.(a) Links espaciais conectados aos eventos na memória episódica; (b) Categorias de objetos referidos pelos links espaciais do evento; (c) Região ocupada pelo agente durante o evento. (a,b) Verdes indicam uma conexão do tipo contexto espacial. Azuis indicam uma conexão do tipo início de evento. Vermelhos indicam uma conexão do tipo final de evento. (c) Cada cor e letra de região refere-se as segmentações adotadas na Figura 6.3.

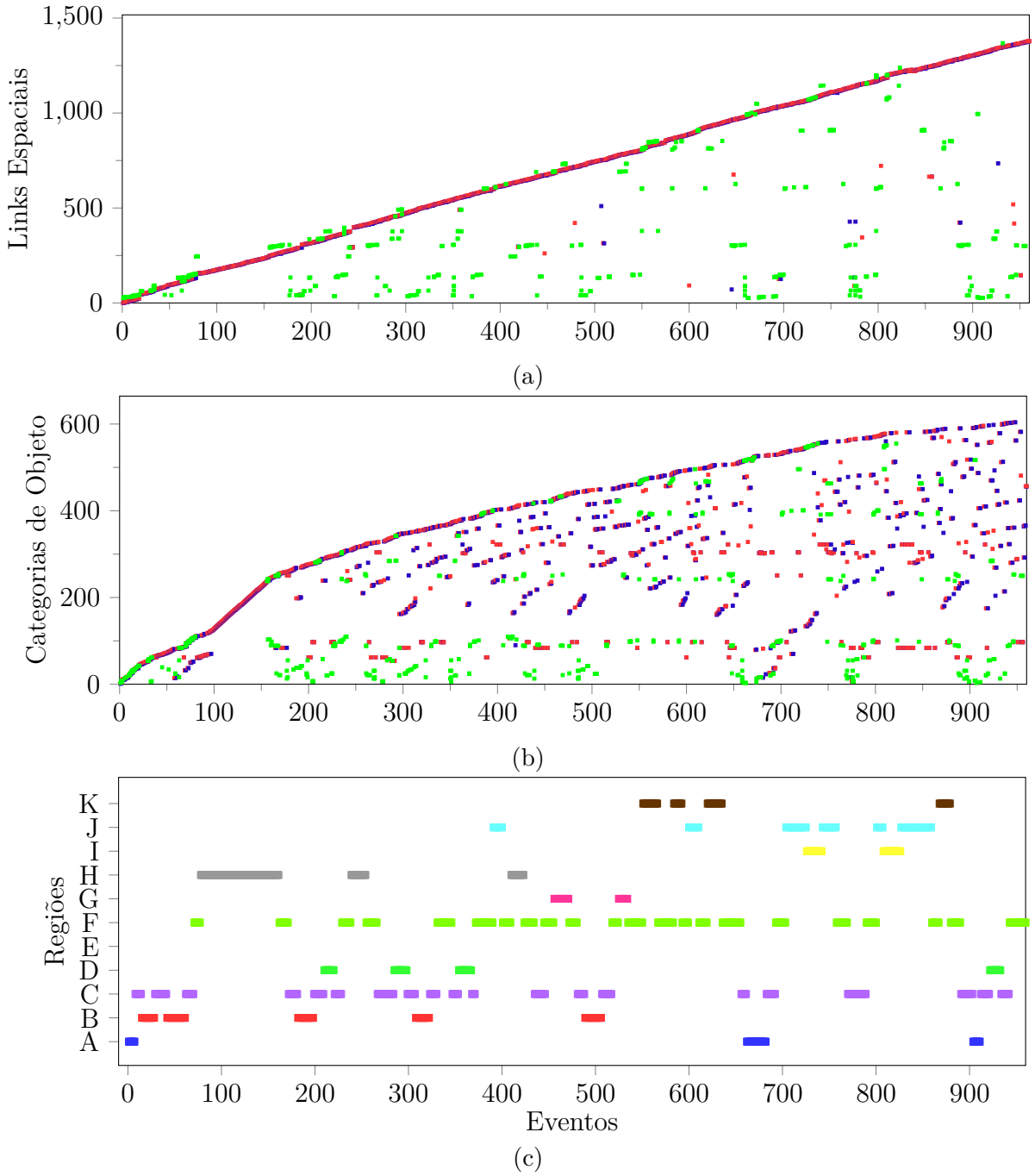


Figura 7.18 – Relação entre nós de eventos e as informações dos links espaciais conectados a eles.(a) Links espaciais conectados aos eventos na memória episódica; (b) Categorias de objetos referidos pelos links espaciais do evento; (c) Região ocupada pelo agente durante o evento. (a,b) Verdes indicam uma conexão do tipo contexto espacial. Azuis indicam uma conexão do tipo início de evento. Vermelhos indicam uma conexão do tipo final de evento. (c) Cada cor e letra de região refere-se as segmentações adotadas na Figura 6.3.

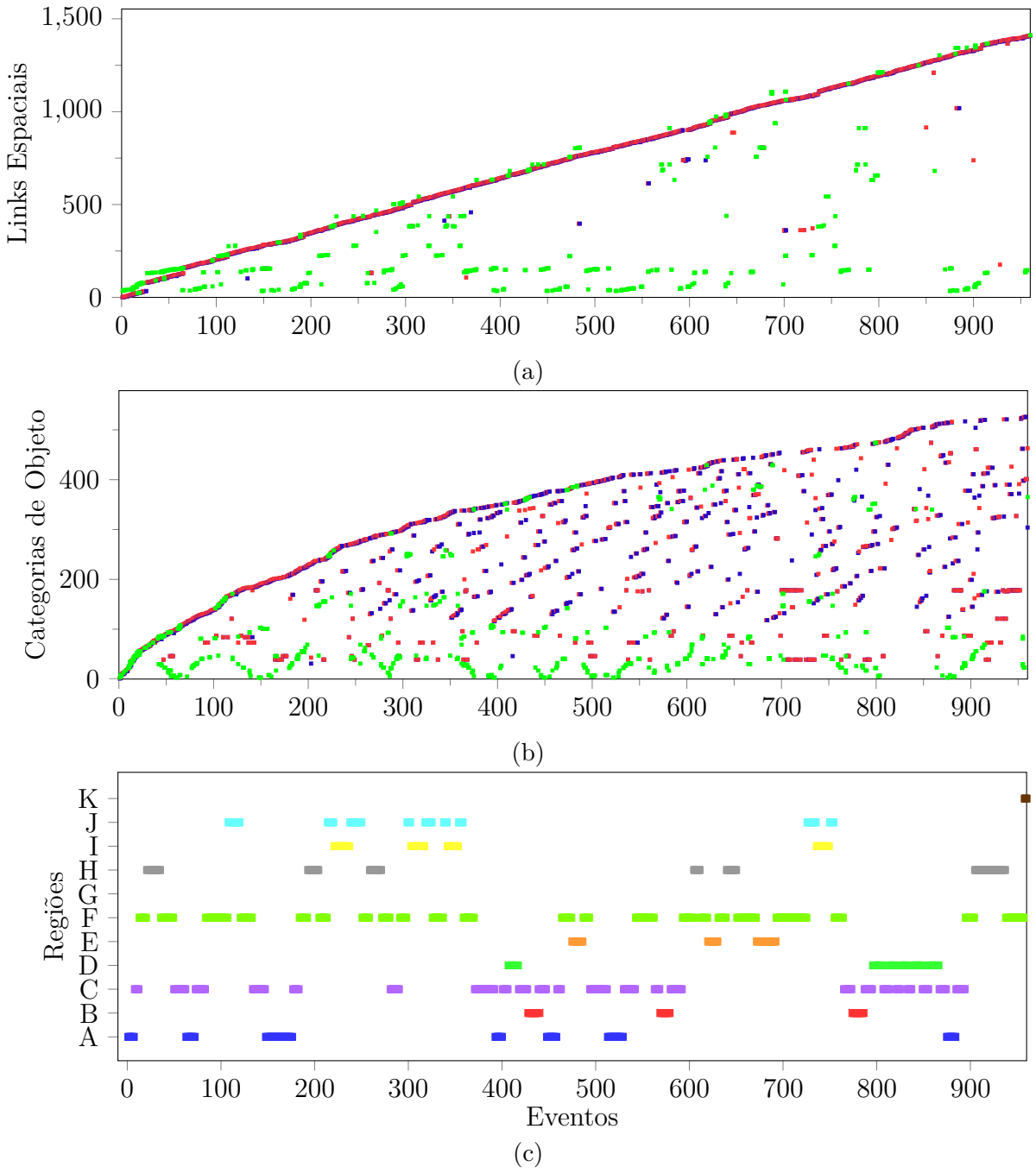


Figura 7.19 – Relação entre nós de eventos e as informações dos links espaciais conectados a eles.(a) Links espaciais conectados aos eventos na memória episódica; (b) Categorias de objetos referidos pelos links espaciais do evento; (c) Região ocupada pelo agente durante o evento. (a,b) Verdes indicam uma conexão do tipo contexto espacial. Azuis indicam uma conexão do tipo início de evento. Vermelhos indicam uma conexão do tipo final de evento. (c) Cada cor e letra de região refere-se as segmentações adotadas na Figura 6.3.