Escola Politécnica da Universidade de São Paulo
Departamento de Engenharia Mecatrônica e Sistemas Mecânicos

Murilo Tavolaro De Nigris

Modelo Computacional para Memória Episódica e Sensória de Robôs Sociáveis

Orientador
Prof. Dr. Marcos Ribeiro Pereira Barretto
marcos.barretto@gmail.com

São Paulo
2014

MURILO TAVOLARO DE NIGRIS

Modelo Computacional para Memória Episódica e Sensória de Robôs Sociáveis

Trabalho de conclusão de curso apresentado à Escola Politécnica da Universidade de São Paulo para a graduação em Engenharia Mecatrônica

Área de Concentração:
Engenharia Mecatrônica

Orientador:
Prof. Dr. Marcos Ribeiro Pereira Barretto

Ficha catalográfica

De Nigris, Murilo Tavolaro
Modelo computacional para memória episódica e sensória de robôs sociáveis / M.T. De Nigris. – São Paulo, 2014.
p.

Trabalho de Formatura - Escola Politécnica da Universidade de São Paulo. Departamento de Engenharia Mecatrônica e de Sistemas Mecânicos.

1.WEB semântica 2.Inteligência artificial I. Universidade de São Paulo. Escola Politécnica. Departamento de Engenharia Mecatrônica e de Sistemas Mecânicos II.t.

AGRADECIMENTOS

Ao Prof. Dr. Marcos Ribeiro Pereira Barretto por todo o tempo e paciência dedicados para me orientar nesse projeto. E também por ser fonte de inspiração e motivação para querer sempre aprender mais.

Aos meus pais e à minha namorada que me apoiaram durante esse ano todo, sempre me motivando e me ajudando como podiam.

Ao meu coordenador no estágio, Giovanne Teles, por ser paciente e compreensivo com as minhas necessidades de tempo para concluir minha graduação.

À Escola Politécnica e ao Departamento da Mecatrônica pela contribuição à pessoa que me tornei ao longo desses anos.

Resumo

A memória humana pode ser dividida em subsistemas de acordo com suas características essenciais. Sendo a memória episódica responsável pela habilidade exclusiva dos seres humanos de reviver experiências passadas e construir uma memória autobiográfica. Existem dois subsistemas que apresentam uma relação direta com esta, memória sensória, responsável por armazenar informações captadas pelos sentidos, como imagens, e sons, e a memória semântica, responsável por armazenar conceitos que representam o conhecimento sobre o mundo. O objetivo deste trabalho é definir e implementar um modelo computacional para a memória episódica e sensória para memória humana pode ser dividida em subsistemas de acordo com suas características essenciais. Sendo a memória episódica responsável pela habilidade exclusiva dos seres humanos de reviver experiências passadas e construir uma memória autobiográfica. Existem dois subsistemas que apresentam uma relação direta com esta, memória s robôs sociáveis. O foco está no suporte que as memórias episódica e sensória fornecem à aquisição de conhecimento, durante o diálogo do robô com um único interlocutor. Com este foco no diálogo, torna-se necessário que a memória sensória armazene imagens estáticas em baixa frequência de amostragem (na ordem de 5 imagens por segundo) porque as imagens serão utilizadas apenas para a identificação de emoções na face e da identidade do interlocutor, bem como todas as vocalizações do interlocutor, em sua forma audível e textual. Ainda, com o mesmo foco no diálogo, interessa que a memória episódica consiga armazenar as informações semânticas associadas ao interlocutor ("quem"), ao assunto em questão no diálogo ("o que"), e à emoção reconhecida. Apesar do projeto prever a adição de funções para tratar os inputs textuais e verbais, foram implementados somente os métodos para tratar reconhecimento de faces e emoção. Além disso, a construção do significado semântico, ou seja, a conexão do "quem" e do "o que" com o conteúdo da memória semântica, é parte essencial do modelo a ser implementado. Assim,a construção de um modelo inicial de memória semântica torna-se parte necessária do presente trabalho. A implementação será baseada no uso do bancos de dados semântico da Oracle, utilizando a API do Jena para interfaceamento entre a aplicação e os modelos criados na base. A representação do conhecimento na memória semântica será construída utilizando-se a linguagem OWL-DL. As atividades de construção da memória autobiográfica visam ser usadas em módulos que implementem uma arquitetura de multi-agentes, permitindo assim a criação de diversas instâncias para ler e salvar informações na memória.

Palavras-chaves: memória episódica, robôs sociáveis, Jena, OWL, banco de dados semântico, memória autobiográfica

Abstract

Human memory can be divided into subsystems according to their essential characteristics. Being the episodic memory responsible for human's unique ability to revive past experiences within their minds and build an autobiographical memory. There are other two subsystems directly related to episodic memory: sensory memory, responsible for storing information captured by the senses, such as images and sounds; and semantic memory, responsible for holding concepts that represent knowledge about the world. The objective of this work is to define and implement a computational model for sensory and episodic memory for social robots. The focus is on the support that episodic and sensory memories supply to the acquisition of knowledge during the dialogue robot with a single interlocutor. With this focus on dialogue, it is necessary that the sensory memory store static images in a low sampling rate (on the order of 5 frames per second) because the images are used to identify emotions in the face and the identity of the caller, as well as all the vocalizations of the interlocutor in his audible and textual form. Still, with the same focus on dialogue, is essencial that episodic memory has the capability to store the semantic information associated with the caller ("who"), the subject matter in the dialogue ("what"), and emotion recognition. Although the project foresee the addition of functions to handle textual inputs and vocals it was only implemented the methods for treating facial recognition and emotion. The construction of semantic meaning, for example, the connection of the "who"and "what"with the contents of semantic memory, is an essential part of the model to be implemented. Thus, the construction of the initial model semantic memory becomes necessary part of the present work. The implementation is based on the use of Oracle's semantic database, using the Jena API for interfacing between the application and the models created in the database. The representation of knowledge in semantic memory is built using OWL-DL. The functions to build the episodic memory along with the sensory memory were designed to be used by modules developed in a multi-agent architecture, thus allowing the creation of multiple instances to read and write information in memory. So every new module added to the robot can use the same repository to acess information.

Keywords : episodic memory, sociable robots, Jena, OWL, semantic database, autobiographical memory

Lista de ilustrações

Lista de Siglas

- **OWL** = Web Ontology Language
- **RDF** = Resource Description Framework
- **JADE** = Java DEvelopment Framework
- **JESS** = Java Expert System Shell
- **STM** = Short Term Memory
- **LTM** = Long Term Memory
- **GER** = General Event Representation
- **URI** = Uniform Resource Identifier
- **URL** = Uniform Resource Locator
- **W3C** = World Wide Web Consortium
- **OWL-DL** = Web Ontology Language - Description Logic
- **SPARQL** : acrônimo recursivo SPARQL Protocol and RDF Query Language

Sumário

1 Motivação

A tecnologia está cada dia mais presente nas nossas vidas, não só em termos quantitativos, mas também de forma qualitativa, com sistemas cada vez mais complexos cujo objetivo é tornar nossa experiência com o mundo mais fácil e mais agradável.

Nos últimos anos tem-se buscado construir robôs, seja com uma forma física ou somente sistemas virtuais, que interajam com seres humanos através de diálogos. Um exemplo disso é a Siri, um programa para celular que conversa com pessoas, e que muitas vezes soa extremamente natural. A Siri recebe comandos de voz e os interpreta para realizar alguma ação dentro do seu celular. Por exemplo, você pode perguntar ”Siri, tem alguma hamburgueria aqui perto?”, ele irá buscar hamburguerias próximas a sua localização, usando um mapa e a informação do seu GPS e responder ”Encontrei alguns lugares próximos.”Em algumas situações, você pode continuar a conversa, e ela usa a informação anterior para compreender o que foi dito.

Outro exemplo disso é o Google Now, de uma empresa concorrente a da Siri, que também recebe comandos de voz e realiza ações no seu celular, caso o sistema não conheça, ele se utiliza da Internet para buscar a informação. Além disso, esse sistema fornece informações personalizadas sem você ter que pedir, como a previsão do tempo, restaurantes similares aos que você frequenta próximos a sua localização atual, o motivo do trânsito no qual você está preso, por exemplo um acidente.

Ambos os sistemas armazenam informações sobre as pessoas obtidas ao longo do tempo e se utilizam delas para ajudá-las no dia-a-dia. Mas mesmo esses sistemas mais complexos são extremamente limitados se comparados à capacidade humana de interagir em diálogos. Os seres humanos realizam tarefas muito complexas de maneira inconsciente como o processamento de falas, rostos e emoções. São capazes de inferir sobre o que é experienciado e guardar essas experiências para serem revividas

posteriormente como lembranças. Tais sistemas, tentam simular essa capacidade de inferência e de uso de informações anteriores para tornar a interação mais humana, porém não se comparam à companhia humana. A motivação desse trabalho está no desejo de permitir que esses sistemas sejam capazes de armazenar as informações provindas de diálogos com humanos de modo que possam ser usadas para construir uma interação a longo prazo, que seja similar ao que existe entre os seres humanos. Existem alguns filmes que ilustram essa capacidade em robôs, um deles é o "Homem Bicentenário", no qual os robôs, que possuem forma humana, são assistentes pessoais, aprendendo com suas experiências diárias e construindo uma memória autobiográfica. Outro exemplo está no filme "Ela", no qual foi desenvolvido um sistema operacional para computadores que possui personalidade única e além de ser capaz de aprender com as experiências diárias, possui a capacidade de reviver essas experiências, ou seja, de modo consciente relembrar de uma situação passada, tanto de quando e onde ocorreu, algo que se acredita ser uma característica exclusiva dos seres humanos.

2 Objetivo

O objetivo do projeto é desenvolver um modelo computacional para memória episódica e sensória de robôs sociáveis, que os permita construir sua memória autobiográfica. O modelo precisa que todas as informações armazenadas estejam descritas e relacionadas segundo regras formais da linguagem OWL para serem processadas por computadores, permitindo o uso de linguagens de programação em lógica para realizar inferência sobre as Ontologias.

Além disso, a implementação desse sistema visa permitir seu uso em robôs cuja construção seja modular, ou seja, módulos que realizam tarefas específicas podem ser desenvolvidos em momentos diferentes e se utilizarem desse mesmo modelo para obter e armazenar informações na memória do robô.

Uma terceira característica desejada é permitir que o conhecimento do robô sobre o mundo possa ser ampliado a qualquer momento sem a necessidade de se apagar ou alterar as informações obtidas em suas interações passadas.

3 Revisão Bibliográfica

3.1 Modelo de Memórias de Seres Humanos

A memória episódica é um sistema hipotético do cérebro que armazena as experiências passadas e permite aos seres humanos, em qualquer momento futuro, revivê-las, sendo também responsável pela construção da memória autobiográfica.[2]
Esse conceito foi proposto como um sistema teórico separado, por Endel Tulving na década de 70. E desde então vem sendo discutido e aprimorado através de experimentos e discussões entre pesquisadores da área.
Acredita-se que esse sistema evoluiu a partir da memória semântica, a qual armazena todo o conhecimento adquirido durante a vida , como significados e conceitos. Mas desenvolveu características únicas, que tornam sua distinção importante.[2] Dentre todos os tipos de memórias, ela é única que apresenta uma relação com o tempo, permitindo que as experiências passadas sejam revividas dentro da mente, sem que sejam confundidas com o momento atual.
Essa capacidade da memória episódica está relacionada a existência de três conceitos, um "eu"próprio, a noção de tempo subjetivo e uma consciência autonoética [2].
Essa última é um tipo especial de consciência que permite ter a noção de tempo subjetivo [2], o qual é necessário para entender que as experiências passadas não fazem parte do tempo atual, e tudo que é revivido está acontecendo apenas dentro da mente que está fazendo essa viagem ao passado.
De acordo com esse modelo, a memória episódica "não é uma tarefa específica nem somente um tipo de experiência mental"[2] mas sim um sistema que apresenta tanto características exclusivas como algumas que são compartilhadas por outros.

Essas características exclusivas também são propostas no modelo de Conway [3], porém este diverge em certos aspectos do modelo de Tulving. Nesse artigo é proposta a separação entre memória episódica e memória autobiográfica. A primeira é constituida por SEM (*Simple Episodic Memory*, em português Memória Episódica Simples), que é constituído por um elemento episódico e um quadro conceitual associado a ele [3]. E diversos SEM constituem um CEM (*Complex Episodic Memory*, em português, Memória Episódica Complexa).

Os elementos episódicos apresentados por Conway são diferentes dos propostos por Tulving, uma vez que são elementos da representação episódica[3] e não elementos do sistema de memória episódica. Sendo armazenados resumos dos eventos, tendo estes início e fim de acordo com seu contexto. Uma vez que o contexto do evento mudou a continuação dele começa a ser armazenado em outro elemento, que terá outro quadro conceitual com uma nova interpretação pessoal sobre o ocorrido.

São esses quadros conceituais que contribuem para que a memória se torne de longo prazo, já que os seres humanos esquecem rapidamente eventos aos quais não foram atribuídos significados, e que não foram considerados relevantes.

Acredita-se também que a memória episódica está ligada à necessidade dos animais de atingir objetivos [3], sendo necessário lembrar do que já foi feito para saber quais são as próximas tarefas, e assim perceber quando o objetivo foi alcançado. Assim, os elementos episódicos que tem relevância para objetivos de longo prazo tem seus acessos mantidos por um tempo maior, até que tal meta seja cumprida.

Em ambos modelos, acredita-se que muitas das tarefas diárias dos seres humanos envolvem mais de um sistema de memória[2]. Por exemplo, no artigo [3] são citadas nove propriedades da memória episódica, porém acredita-se que algumas delas estejam relacionadas também a memória sensória ou a outros tipos de memória, porém somente a episódica possui as nove [3]. As principais características seriam, manter resumo dos registros sensórios, possuir uma perspectiva, representar curtos períodos

da experiência, estarem representados numa linha do tempo, tornar a recordação de momentos autobiográficos específica e serem experiências revividas.
A memória sensória citada anteriormente é responsável por captar informações (imagens, sons, cheiros, etc.) do ambiente externo através dos sentidos, acredita-se também que ela trabalhe em períodos muito mais curtos que a memória de curto prazo, na ordem de menos de um segundo e de menos de um minuto respectivamente.

3.2 Trabalhos Relacionados

Um exemplo de um trabalho que utiliza a idéia de ter uma memórias semântica separada para armazenar ontologias é o ORO (*Open Robots Architecture*), que é uma plataforma de serviço de conhecimento baseado em Ontologias. Ele permite que "outros módulos se conectem nele inserindo ou buscando conhecimento inferido de um repositório central"[4]. A estrutura utilizada para armazenar as informações é baseada na linguagem RDF e desenvolvida utilizando a API do Jena e seu conjunto de ferramentas.
Para ampliar o conhecimento semântico podem ser conectadas diversas ontologias, na linguagem OWL(*Ontology Web Language*), ao backend do modelo. E para a realizar inferências sobre as ontologias foi desenvolvida uma integração com a ferramenta Pellet, um framework para descrição lógica feito para ser utilizado com a linguagem OWL.
O ORO permite que sejam realizadas buscas usando a linguagem SPARQL, buscas por conceitos em diversos idiomas e evita que novos fatos inconsistentes sejam adicionados. Nesse artigo [4], o ORO é utilizado em um robô com reconhecimento visual do cenário onde está inserido, cujo objetivo é aprender sobre um objeto desconhecido e armazenar esse novo conceito na sua memória, de maneira integrada com seu conhecimento anterior.
Outra implementação similar foi realizada no artigo [5], que apresenta um

modelo para conhecimento semântico, utilizando o Jena como framework para trabalhar com esse conhecimento, e o JADE (*Java Agent DEvelopment Framework*) para criar uma estrutura de multi-agentes. Além disso, é utilizado o JESS (*Java Expert System Shell*), uma linguagem de programação em lógica assim como o Pallet, para criar regras que permitam a inferência sobre o conhecimento.
Nesse trabalho o programa Protégé e sua API com o JESS também são utilizados para construir e editar as ontologias usadas na linguagem OWL. Primeiro foi criada uma ontologia simples na interface do Protége e depois ela foi expandida usando a API que permite programar na linguagem JESS, criando regras para inserir fatos na Ontologia.
No artigo é mostrado um exemplo, no qual na Ontologia criada, cada marca de carro possui sete modelos, e cada um deles possui três atributos: nome, preço e o estado de conservação.[5] Para trabalhar com o conhecimento desse modelo são implementados dois agentes, o primeiro *AskAgent* ("Agente que pergunta"), responsável por transformar a pergunta do usuário em uma *query* no formato SPARQL, e passá-la para outro agente utilizando o protocolo ACL de comunicação (característica importante do framework JADE). O segundo agente, *AnswerAgent* ("Agente de resposta") executa a *query* transformando o resultado no formato RDF e enviando de volta a informação usando, também, o protocolo ACL.
Os dois trabalhos acima citam modelos computacionais para uma memória semântica separada do resto do sistema, que qualquer outro módulo possa utilizar. Porém não foi implementado nada similar à memória episódica tanto de Tulving como de Conway.
Já o artigo de [6], apresenta um modelo similar ao de Conwway[3], no qual se propõe a criar um modelo genérico para a memória de robôs sociáveis, o qual seja adaptável para cada ambiente e que possa ser utilizado por um longo período de tempo.
O modelo apresenta três módulos principais, a memória de curto prazo STM (*Short Term Memory*, em português Memória de Curto Prazo),

memória de longo prazo LTM (*Long Term Memory*, em português Memória de Longo Prazo) e o conhecimento sobre o mundo WK (*World Knowledge*, em português, Conhecimento sobre o Mundo), que é equivalente à memória semântica proposta anteriormente.
Dentro do modelo apresentado, são importantes: o módulo GER (*General Event Representation*, em português, Representação de Eventos Genéricos) no qual os objetivos são formulados e categorizados em tempo real com a ajuda da memória ontológica (semântica), permitindo que possamos separar aqueles relacionados a compromissos; e módulo LTM que é responsável por monitorar constantemente o status do objetivo, que uma vez cumprido tem essa informação atualizada e a meta fica armazenada no módulo GER. Além disso, esse trabalho evita ações que possam violar as metas, consultando a memória autobiográfica.
Já o trabalho proposto em [7] separa a memória de longo prazo em memória episódica e semântica, ao invés de considerar as duas como tipos distintos. O objetivo apresentado é desenvolver um robô sociável para ser utilizado em um ambiente de hotel, realizando diversas tarefas, entre elas, acompanhar pessoas até os quartos, dar informações turísticas, recolher pedidos, engajar-se em diálogos, entre outras.
A memória semântica, que é responsável por armazenar o conhecimento do robô, está estruturada na forma de Ontologias, criadas na ferramenta Protégé[7] pela equipe do projeto contendo informações relevantes ao ambiente de hotel no qual o robô está inserido.
Além da memória semântica, é modelada a memória episódica para armazenar interações passadas do robô, que são organizadas na forma de casos. Esse formato foi escolhido com o objetivo de resolver problemas usando o processo CBR (*Case Base Reasoning*, em português Raciocínio Baseado em Casos), que ocorrem em quatro passos: *Retrieve* (Recuperar), que busca na memória casos semelhantes ao problema atual, e retorna o mais próximo; *Reuse and Revise* (Reutilizar e Revisar), que recebe esse caso similar, o revisa, adaptando-o para a situação atual; e *Retain* (Reter) que armazena a solução proposta e o caso que foi

resolvido na memória [7]
Para que esse processo funcione são construídas estruturas de conhecimento baseadas no conhecimento contido na memória semântica, assim, os problemas reais a serem resolvidos podem ser transformados para uma estrutura que a arquitetura do robô compreenda e consiga levar para as próximas etapas do processo.
Além disso, os casos acessados na memória episódica podem recuperar outras informações ligadas por uma ordem temporal dos acontecimentos passados.

4 Introdução Conceitual

Neste capítulo serão explicadas as ferramentas e frameworks escolhidos para a realização desse trabalho.

4.1 Resource Description Framework (RDF)

RDF é um estrutura para descrever os recursos da Web, sendo uma das formas de representação da OWL, ou seja, uma linguagem contendo a sintaxe básica com a qual a OWL é escrita e pode ser transmitida (não sendo o único formato, já que a OWL pode ser distribuída no formato XML também).

O RDF representa seus recursos através de URIs (*Uniform Resource Identifier*, em português Identificador Uniforme de Recursos), que é uma cadeia de caracteres usadas (*string*) para identificar ou denominar um recurso na Internet, sendo as *Uniform Resource Locator* (URL, em português, Localizador Uniforme de Recursos), comumente usadas como endereços de sites, um de seus tipos. Porém as URI não necessariamente são endereços para os conteúdos da Web, e sim possuem uma sintaxe própria para identificar os recursos.

O modelo RDF utiliza as chamadas Triplas, que são afirmações, para representar as relações entre as informações, sendo baseadas em: sujeito, predicado e objeto.[8] Por exemplo, existe um recurso que representa a pessoa eu, onde o nome completo é João da Silva e o título é Dr., sendo esse recurso a URI <http://www.w3.org/People/EM/contact#me> [9]. A figura 1 representa esse exemplo.

Dessa forma a estruturação em RDF fica:

<http://www.w3.org/People/EM/contact#me>
<http://www.w3.org/2000/10/swap/pim/contact#fullName>
”João da Silva”.

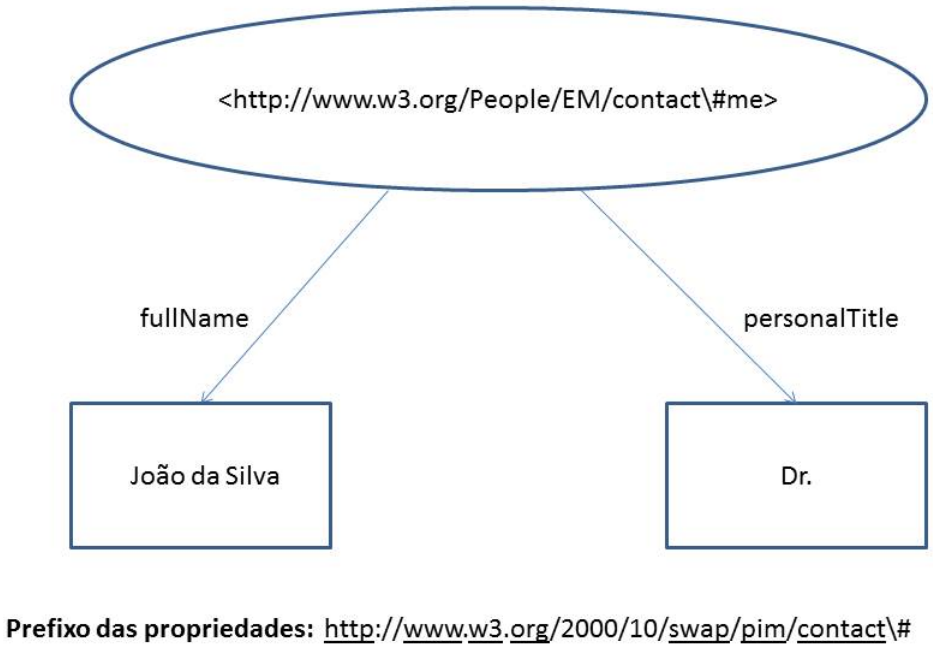

Figura 1 – Exemplo RDF

<http://www.w3.org/People/EM/contact#me>
<http://www.w3.org/2000/10/swap/pim/contact#personalTitle>
”Dr.”

Existem,também, duas outras formas de se representar as triplas, o formato Turtle e o RDF/XML, que são implementados de maneira semelhante mas com uma estrutura diferente.

Como essas representações foram feitas para compreensão dos computadores, documentos grandes nesse formato não são simples de serem compreendidos pelos humanos, assim, existem ferramentas que nos permitem traduzir esses formatos de volta para a linguagem usual, como por exemplo, o software Protégé, que será mostrado em detalhes mais a frente.

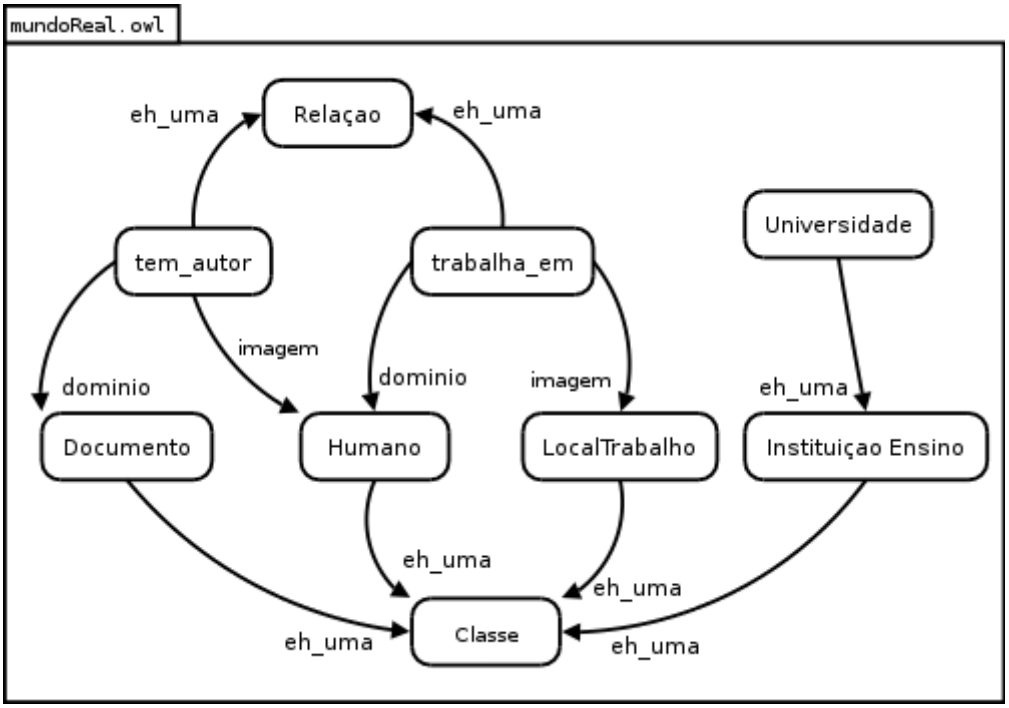

Figura 2 – Diagrama de Ontologia OWL, retirado de [1]

4.2 Ontologia - OWL

A Internet que conhecemos possui todo o seu material organizado em uma forma que os humanos compreendem, mas os computadores não. Para mudar isso existe a Web Semânica, idealizada pelo *World Wide Web Consortium (W3C)*, cuja idéia é permitir que os computadores possam nos ajudar a encontrarmos informações de maneira eficiente, respondendo às nossas solicitações de acordo com seus significados e não só padrões simbólicos[5].
Com esse objetivo o W3C criou a *Web Ontology Language* (OWL, em português Linguagem de Ontologia da Web), cuja proposta é a descrição rica e complexa dos recursos da Internet e suas relações[10].
Os documentos OWL são também conhecidos como Ontologias, podendo ser publicados na internet, e se conectar criando Ontologias mais complexas que são capazes de se complementar. A figura 2 mostra um exemplo básico de como uma ontologia é estruturada.
A linguagem OWL permite a existência de:

· **Classes:** onde todas são subclasses de *Thing* (em português, Coisa),

como por exemplo na figura 2 que temos Documento, Humano, LocalTrabalho e InsitiuiçãoEnsino como Classes.

· **Propriedades de Objeto:** que permitem relacionar dois objetos (nesse caso são instâncias das classes). Nas triplas explicadas anteriormente, essas propriedades são um dos tipos de predicados, sendo assim, ela conecta um sujeito a um objeto, mas a OWL nos permite definir qual classe de sujeito e de objeto podem ser relacionados. Na figura 2, existem as relações tem_autor, que só permite sujeitos da classe Documento e objetos da classe Humano, e trabalha_em, que liga Humano a LocalTrabalho.

· **Propriedades de Dado:** é o segundo tipo de predicado existente, ele conecta um sujeito, que pertence à uma classe e um objeto que é literal, que abrange os tipos básicos de dados, como inteiros, *long*, String, entre outros, mas não é de nenhuma classe.

· **Indivíduos:** que são instâncias das Classes citadas acima. Cada indivíduo pode pertencer a 0 ou mais classes, sendo um pouco diferente do modo como é usado em orientação a objetos.

Com essas quatro categorias de dados, e todas as operações, como união, complemento, intersecção, simetria, transitividade e muitas outras é possível representar sistemas complexos e que modelam com detalhes alguns domínios da realidade, como por exemplo a genealogia, é possível representar completamente a relação entre todas as pessoas de uma família.

Além disso a OWL possui três sub-linguagens, uma delas, a OWL-DL, que é baseada em lógica de descrição (DL - Description Logic), fornece máxima representação garantindo total vínculo das informações e término em tempo finito das execuções de tarefas de inferência[11], sendo ideal para sistemas que necessitam dessas tarefas.

Outra sub-linguagem é a OWL Lite que apresenta uma simples hierarquia de classificação com algumas características de restrição, sendo

recomendada para modelos simples que exigem uma migração rápida de outros formatos. Enquanto que a OWL Full possui máxima representatividade porém não garante que todas as características sejam possíveis de serem utilizadas pelos programas baseados em descrição lógica.
Para esse projeto foi escolhida a OWL-DL, pois é necessário que o modelo de memória seja capaz de processar qualquer relação existente nele, e também porque as ontologias que podem ser adicionadas para ampliar o conhecimento do robô podem vir a ser muito mais completas do que a OWL Lite permite.

4.3 Oracle Semantic Graph Database and Jena API

Para usar a linguagem OWL no projeto foi necessário encontrar um banco de dados que fosse compatível com as principais propriedades da linguagem, dentre as opções pesquisadas foram encontradas três opções principais o Virtuoso, o Jena TDB, e o Oracle Semantic Grapg Database. O Jena TDB e o Oracle possuíam a vantagem da compatibilidade com a API Jena, que encapsula as funções que acessam diretamente os nós e arestas dos grafos, permitindo trabalhar com modelos em uma linguagem de mais alto nível. Essa API apresenta duas vantagens, a primeira pois diminui o número de passos para realizar determinada tarefa, e a segunda pois permite que código seja desenvolvido usando diretamente as classes e objetos que são propostos para o modelo (serão explicados na próxima seção).
Entre esses dois, o Oracle foi escolhido por ter uma documentação bem detalhada de todo seu funcionamento e propriedades, bem como permitir trabalhar com uma arquitetura distribuída caso seja necessário devido ao crescimento acelerado que o tamanho da memória possa vir a ter.
Além disso, os três bancos são compatíveis com o SPARQL, uma linguagem utilizada para fazer *queries* tanto de busca como inserção em bancos de dados orientados a grafos semântico. Apesar da API do Jena

implementar algumas funcionalidades do SPARQL, não necessitando escrever as queries manualmente, é possível usá-las diretamente caso seja necessário, tanto por motivos de extrema complexidade como para manter o sistema o mais genérico possível.
A configuração e estrutura usado na banco será explicada depois que forem definidas como ficarão as classes da aplicação nos capítulos seguintes.

4.4 JADE

JADE (Java Agent DEvelopment Framework) é um middleware implementado em linguagem Java que obedece às especificações da *The Foundation for Intelligent Physical Agents* (FIPA [12] , em português "A Fundação para Agentes Físicos Inteligentes") cujo objetivo é permitir sua utilização em arquiteturas distribuídas de multi-agentes. Para garantir a operação dos agentes o protocolo Agent Communication Language (ACL, em português "Linguagem para Comunicação de Agentes"), da FIPA é utilizado.
Além disso, devido ao JADE ser implementado em Java existem outros benefícios, como por exemplo, os sistemas operacionais que contém os agentes não precisam ser os mesmos, e estão disponíveis vários recursos gráficos e ferramentas para "debug".
Uma implementação do JADE foi citada anteriormente no trabalho [5] que sugere um modelo para o conhecimento semântico, onde existem dois agentes, um responsável por realizar uma pergunta, e o outro por fornecer a resposta, buscando-a na Ontologia à qual está conectado.
O agente que envia a pergunta recebe o comando do usuário e constrói a *query* na linguagem SPARQL (que será explicada mais a frente) passando-a para o agente de resposta.
O agente da reposta executa essa *query* na Ontologia a qual está conectado, retornando o conteúdo através de diversas mensagens do protocolo ACL, no formato RDF [5].

5 Arquitetura Proposta

5.1 Definição do Problema

O modelo computacional proposto tem como objetivo ser usado em robôs sociáveis, portanto o foco das funcionalidades desenvolvidas está na construção de sua memória autobiográfica através da participação em diálogos.

Para alcançar esse objetivo, foram estudados alguns modelos existentes para o funcionamento de cérebro humano e proposto um modelo com características apresentadas por Tulving[2] e por Conway[3].

A arquitetura do robô se baseia em três tipos de memória, sensória, episódica e semântica, que serão implementadas separadamente, cada uma com seus métodos e atributos, diferentemente dos seres humanos onde essa divisão é apenas conceitual[2].

Para desenvolver um sistema em robôs que imite a memória humana, é necessário que todas as informações armazenadas possam ser processadas por computadores, ou seja, elas devem estar relacionadas e organizadas através de um conjunto formal de regras que permita a busca semântica e inferência de conceitos não explícitos. E também que os eventos possuam uma relação temporal[2], permitindo reconstruir a experiência na mesma ordem que ocorreu.

Apesar da memória episódica ser a mais importante para conectar as informações e registrar as experiências do robô, pelo menos duas das três memórias serão utilizadas para realizar o processo completo da maioria das tarefas, assim como nos seres humanos. Em termos de implementação, os métodos de cada memória serão chamados separadamente e seguindo uma ordem definida pelos módulos que a utilizarem.

Segundo o modelo apresentado em [3], a memória sensória é responsável por armazenar as informações obtidas através dos sentidos, como visão e

audição. A memória episódica guarda as experiências autobiográficas, ou seja, os eventos que ocorrem são armazenados e identificados através de início e fim e também pelos conceitos genéricos e específicos associados. Já a memória semântica é responsável por armazenar todo o conhecimento do robô sobre o mundo.

Para exemplificar que as Memórias são sempre utilizadas juntas, vamos imaginar que ao conversarmos com alguém conhecido, nosso cérebro captura a imagem de uma pessoa, processa e a identifica. Ao fazer isso, ele também está guardando na memória episódica o evento de encontrar e conversar com tal pessoa em determinado momento. Esse evento, está associado à diversos conceitos retirados da experiência, como o conceito da pessoa com quem conversou, os tópicos que foram abordados, e o conceito de determinada emoção que a pessoa apresentou em tal momento. Por exemplo, se interpretamos que a pessoa estava feliz naquele momento associaremos esse evento ao conceito de felicidade. Além disso, o evento está associado às imagens obtidas e guardadas na memória sensória, já que somos capazes de nos lembrar, por exemplo, da roupa que a pessoa estava usando.

Como definição do projeto, o sistema de memória deve ser capaz de tratar tanto as informações da linguagem, que poderiam ser obtidas por voz ou texto, como as informações visuais. Porém, nesse projeto foram testadas somente funcionalidades para tratar os inputs visuais.

Os agentes para os quais o sistema foi desenvolvido até o momento pertencem ao Módulo Face do Robô Minerva, implementado no trabalho [13].

5.2 Ontologia para representar modelo das unidades de cada memória

Para desenvolver o sistema foi necessário definir como ficariam estruturadas as unidades de cada tipo de memória. A figura 3 mostra a definição que foi escolhida, sendo que os quadrados com balões no meio

representam recursos, ou seja quando o sujeito ou objeto da tripla não é um valor literal, mas sim pertencente a uma ou mais classes da Ontologia, já os balões simples são valores literais.

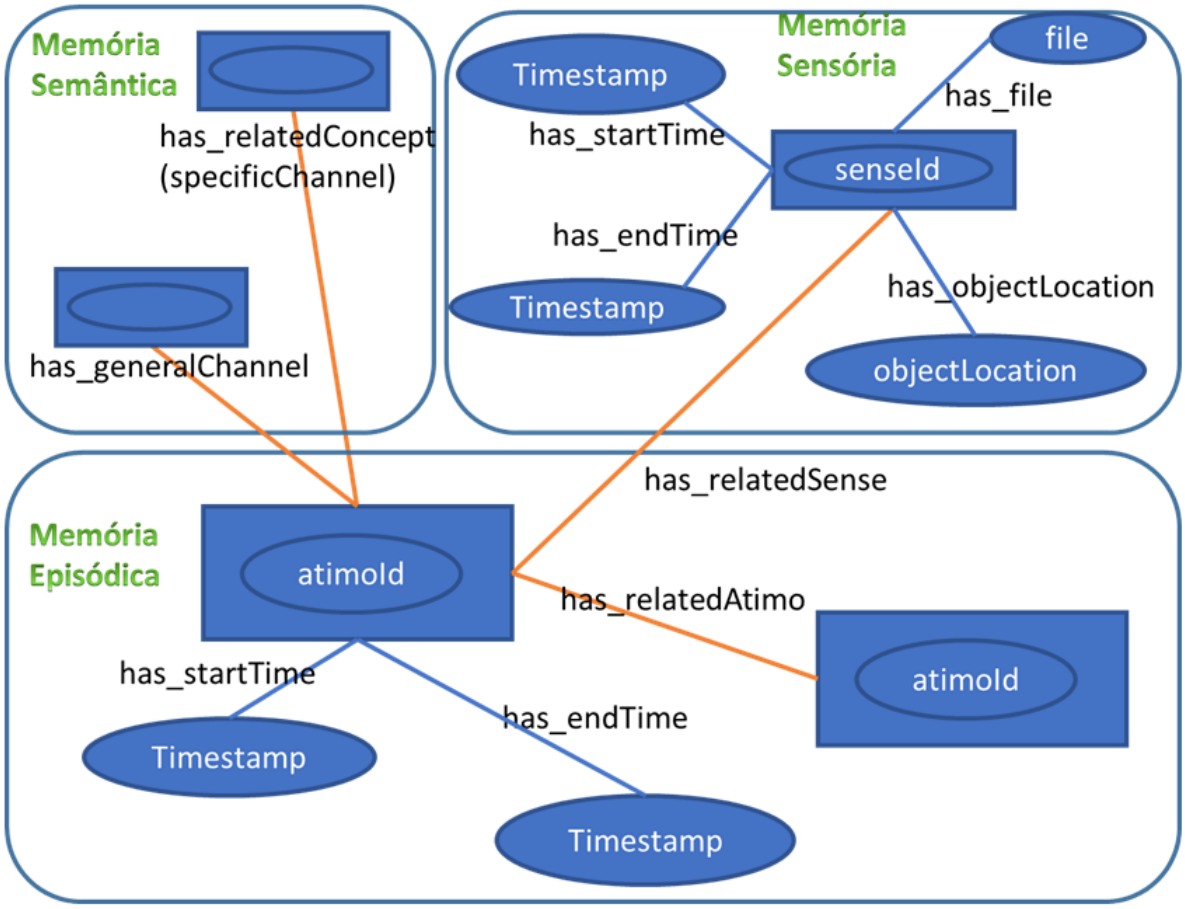

Figura 3 – Modelo para Unidades de Cada Memória

Na memória episódica temos a unidade chamada de Átimo, ela possui seis propriedades, ou seja, é caracterizada por seis triplas. O recurso ”atimoId”, é uma URI que possui um código único, o qual para o programa é um *timestamp*, já que as coisas irão sempre ocorrer ao longo do tempo. Para cada um dos Átimos, há um início e fim, definido por *timestamps* cujo tipo é *long*, existe relações com zero ou mais átimos, sendo que cada relação poderá ser de um tipo pré-definido pelo modelo da Ontologia. Por exemplo, quando o robô conversar com uma pessoa, e reconhecer uma emoção, ele pode salvar dois átimos que ocorrem no mesmo período, um sobre a pessoa o outro relacionado à emoção, e

ambos estariam conectados, podendo ser em uma direção ou nas duas, por exemplo, ”has_emotion”e ”is_emotion_of”. O predicado ”has_generalChannel”caracteriza qual dos tipos de informação esse átimo representa, no exemplo citado acima, o da emoção estaria associado ao conceito emoção da memória semântica, e o da pessoa estaria associado ao conceito pessoa. Já o predicado ”has_relatedConcept”, no caso da emoção, estaria conectado a qual emoção foi detectada, tristeza, felicidade, raiva entre outras, enquanto que no caso da pessoa, estaria conectado ao conceito dela, por exemplo, João de Souza. E por fim há o ”has_relatedSense”, que conecta o Átimo ao dado externo que foi capturado, no caso do uso pelo Módulo Face, serão sempre imagens.
Na memória sensória estarão salvas mais informações referentes a cada imagem, mas o Átimo se conecta somente ao recurso que possui o Id do Sense. Essa estrutura é similar a do Átimo, cada uma das propriedades estão definidas pelas respectivas triplas. O Sense (unidade básica da Memória Sensória) possui um início e fim, onde também serão usados *timestamps* do tipo *long*. O predicado ”has_file”conecta a um arquivo, que até esse momento será somente imagem, mas depois poderá tanto ser voz, como texto. Além desse há o ”has_objectLocation”, que armazena as localizações das faces encontradas na imagem.
Para que tudo isso possa funcionar é necessário criar a Ontologia onde são declaradas as classes às quais cada Recurso pertence, e criar tanto as propriedades de objetos, como as propriedades de dados. Essa Ontologia foi feita no software Protégé e será inserida na memória semântica, onde ficará armazenada junto com todo o conhecimento disponível. Isso é importante para que o robô possa realizar inferência sobre todos os três tipos de memória, tanto como modelos separados como um modelo feito pela união de todas.
Como foi explicado na seção sobre OWL, o modelo de Ontologia criado possui suas regras e definições, e cada nova informação inserida é um indivíduo que deve se adequar às propriedades pré-definidas, ou seja, toda a informação só será aceita se for coerente, o que contribui para manter a

consistência dos dados.

5.3 Diagrama de Classes

Uma vez que foram definidas as estruturas das unidades de cada tipo de memória, algo importante para a criação dessa ontologia, é necessário estruturar como a aplicação irá executar os casos de uso. Para isso foi feito o Diagrama de Classes mostrados nas figuras 4, 5, 6, o sistema foi dividido em dois pacotes, *memoryElements*, que contém classes de objetos especiais que precisam ser armazenados em um formato específico e que possam ser alterados facilmente. A classe *Image*, *ObjectLocation* e *BoundingBoxImplementation*, assim como a *Interface BoundingBox* estão dentro desse pacote. A primeira, mostrada na figura 4, recebe um array de *bytes*, para que possa ser armazenada dessa forma no banco de dados, utilizando o formato BLOB (*binary large object*, em português Grande Objeto Binário). Já o *ObjectLocation*, possui uma lista de *BoundingBox*, e uma *String* para identificar o que está sendo localizado, como por exemplo faces ou mesmo objetos. Já essa interface representa objetos que contém coordenadas do que foi localizado nas imagens.
Dessa forma, pode ser integrado ao robô qualquer módulo capaz de detectar formas específicas, permitindo que a capacidade de reconhecimento seja ampliada gradativamente através de vários projetos desenvolvidos de maneira independente.
Dentro do pacote *memorySystem* há a classe *MemoryManager* que é responsável por controlar as instâncias de acesso ao banco de dados e a utilização dos modelos, que são os três tipos de memória: sensória, episódica e semântica. Essa classe garante que todos os agentes terão acesso a todas os modelos de memórias através de uma única instância.
No *memorySystem* também foram criadas as classes que representam as unidades da memória sensória e episódica, *Sense* (e *VisualSense*) e *Atimo* (palavra utilizada no lugar de ”evento”, cujo significado é momento, ou instante, e que representa a menor unidade da Memória Episódica),

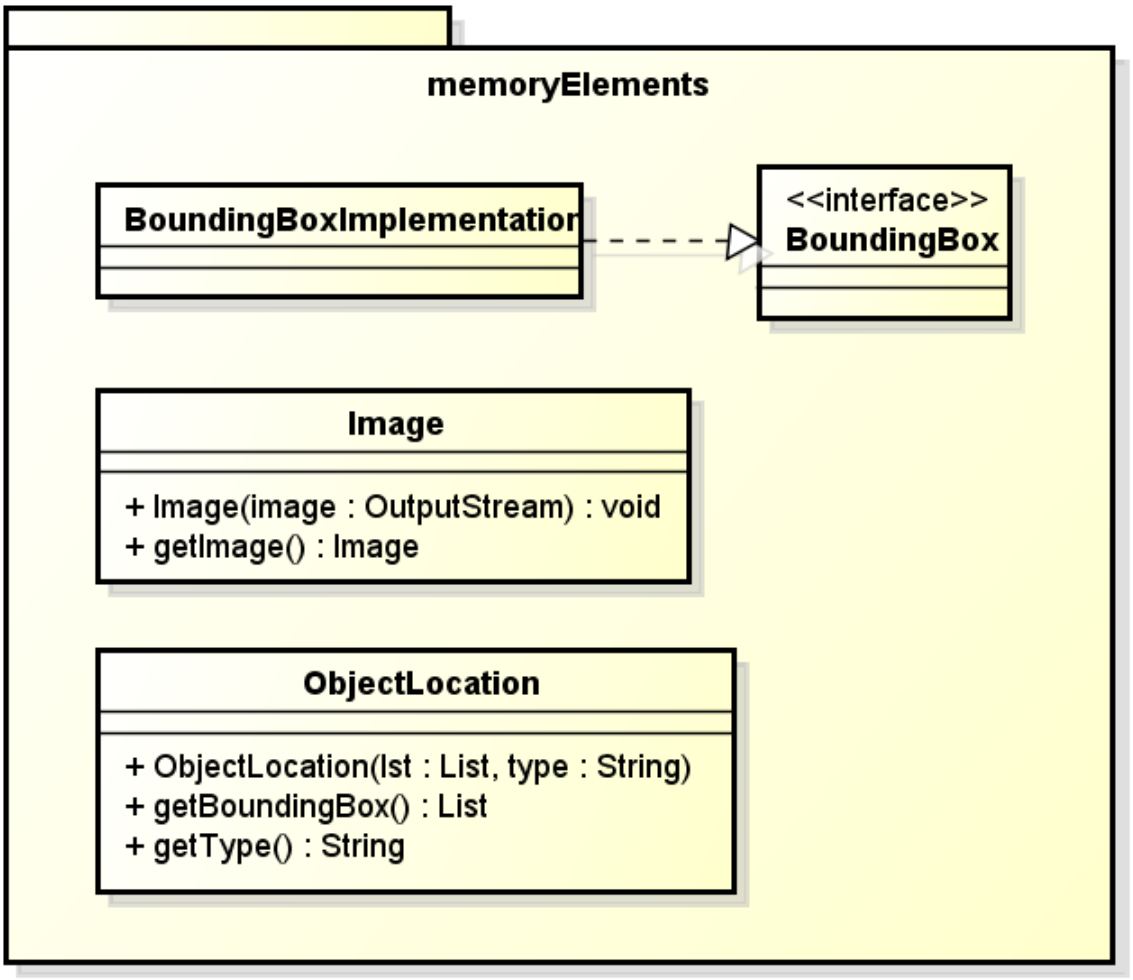

Figura 4 – Diagrama de Classes - Pacote memoryElements

respectivamente.

A classe *Sense*, que contém somente os atributos *startTime*, *endTime* e *senseId*, é superclasse da *VisualSense*, a qual tem sua estrutura mostrada na figura 3. O propósito da criação de uma subclasse é permitir que diversos sentidos possam ser implementados no robô, cada um com propriedades únicas e distintas dos outros.

A classe *Atimo* apresenta os atributos mostrados na figura 3, sendo a principal responsável por conectar as informações gravadas na outras duas memórias. Além disso, os *Atimos* podem se conectar uns aos outros através de tipos de ligações pré configuradas e definidas pelo controle feito na classe *EpisodicMemory*.

O método *StoreAtimo* armazena os *Atimos* no banco de dados controlando sua continuidade. Ele decide se um novo átimo que foi fornecido para ser armazenado é uma continuação ou é novo. Caso seja novo, todas as informações recebidas são armazenadas e conectadas a um *Átimo* com um novo *Id*, porém, se for somente uma continuação, o único valor a ser trocado é o *endTime* (em português, tempo final).
A função para determinar tal informação é um grande diferencial que pode ser obtido usando a arquitetura proposta. Nesse trabalho foi implementado um método simplificado para realizar essa validação. Um átimo é o mesmo desde que seu conceito genérico (*GeneralChannel*) e específico (*SpecificChannel*) sejam os mesmos, e também tenha se passado menos de cinco minutos da última informação recebida.
Na classe *EpisodicMemory* há também o método *retrieveAtimo* para buscar no banco de dados um átimo gravado, baseado em seu Id, construído em um método a parte (*defineAtimoId*), e o método *related* para salvar na base uma relação entre átimos.
No pacote *memorySystem* foram implementadas, também, as classes para a memória sensória e semântica. A primeira possui métodos (*storeVisual e retrieveVisual*, respectivamente para salvar e buscar no banco de dados objetos do tipo *Image* e *Object Location*, que contém as imagens capturadas, e a localização de características detectadas, podendo ser rostos, objetos ou qualquer outra pré-definida. Essa classe permite salvar separadamente uma *Image* e um *Object Location*, já que agentes diferentes podem realizar as funções de capturar imagem e de localizar determinada característica.
A memória semântica apresenta métodos para buscar conceitos existentes no banco de dados (*searchSubject, getResource*), relacionando o tipo de tarefas que estão sendo executadas pelos agentes que utilizam as outras classes, e também um método para adicionar conceito, proposto para ser usado quando for detectado uma nova entidade da característica buscada. Além disso, nesse projeto, como padrão, existe apenas a ontologia sobre a própria estrutura da memória, porém pode ser adicionada qualquer uma,

sobre o domínio de conhecimento desejado. Isso será exemplificado no Capítulo 6.

Todas as informações adicionadas pelas classes da memória devem ter suas transações *"committed"* (ou seja, gravadas de modo permanente) através da execução do método *commit* implementado em cada uma delas. Essa estrutura foi definida devido ao modo como a API do Jena funciona, onde são definidos modelos cada um com suas afirmações (*statements*).

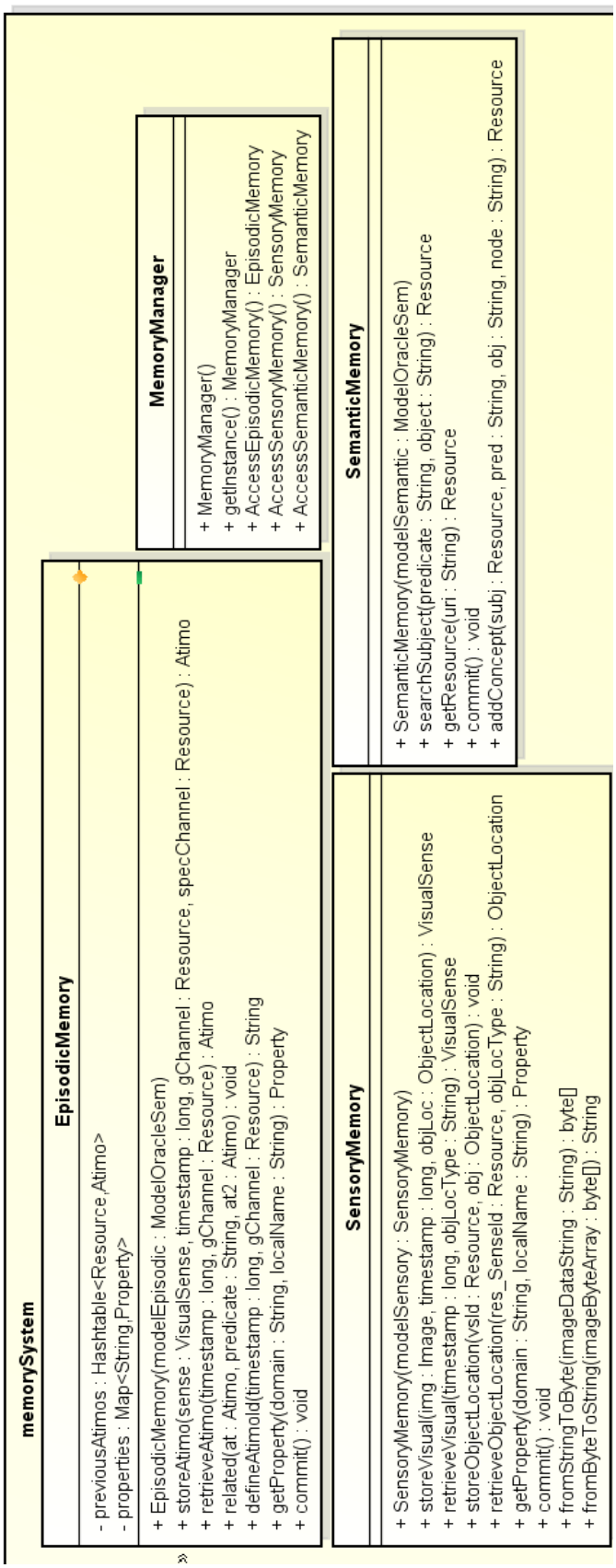

Figura 5 – Diagrama de Classes - Pacote memorySystem - Imagem 1

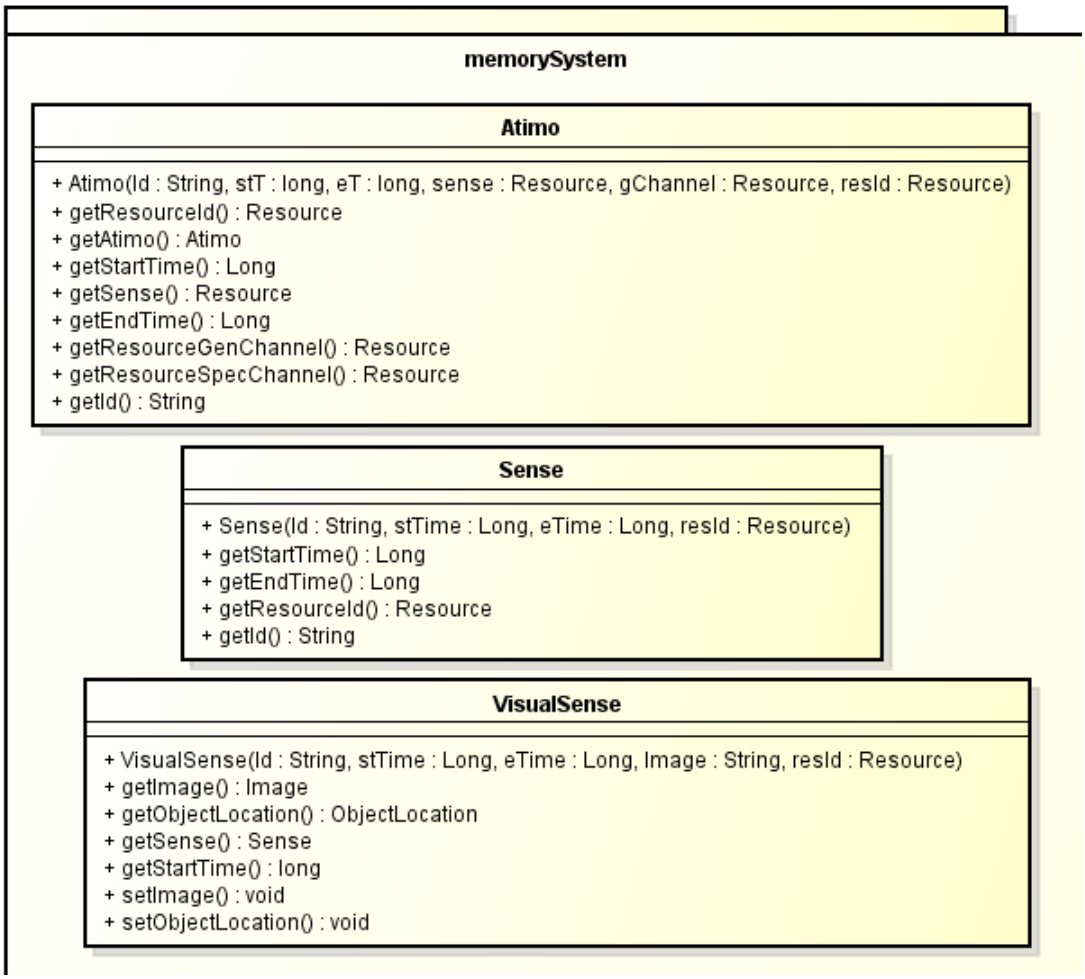

Figura 6 – Diagrama de Classes - Pacote memorySystem - Imagem 2

6 Resultados

6.1 Descrição do Experimento

Para testar o programa que desenvolvido em um caso real, foi escolhido o Módulo Face[13] do robô Minerva[14] para ter seus métodos alterados para utilizar a memória ao invés do armazenamento em arquivos dentro do sistema operacional.

Esse módulo foi desenvolvido utilizando o *middleware* JADE, explicado na seção 4.4, sendo definidos diversos agentes capazes de executar suas tarefas separadamente e se comunicar através de um único protocolo. Os Agentes criados que representam as tarefas de um robô sociável foram *ImageAgent* (Agente Imagem), *DetectionAgent* (Agente Detecção), *PersonRecognitionAgent* (Agente de Reconhecimento de Pessoas), *InstantEmotionRecognitionAgent* (Agente de Reconhecimento de Emoção Instantânea e *ContinuousEmotionRecognitionAgent* (Agente de Reconhecimento de Emoção Contínua), dos quais somente o último não será usado nesse projeto. Existem outros agentes criados que não foram citados aqui pois são para utilizar o Módulo Face em um computador pessoal através de uma interface gráfica onde é possível visualizar os resultados durante a execução, não fazendo parte das tarefas do robô sociável.

Para definir quais métodos devem ser utilizados por cada agente foi necessário fazer uma análise do funcionamento desse módulo, determinando através de um Diagrama de Casos de Uso, as necessidades de cada um.

O *ImageAgent* é responsável por capturar imagens em determinado intervalo de tempo e armazená-las, para que o *DetectionAgent* detecte a localização dos rostos na imagem, caso existam. Já o *PersonRecognitionAgent*, para cada face detectada, irá extrair as características do rosto e procurar qual pessoa já conhecida tem maior

semelhança. E o *InstantEmotionRecognitionAgent*, irá extrair características do rosto através de um algorítimo e comparar com os parâmetros universais de emoção.

O diagrama de Casos de Uso mostrado na figura 7 define o que o cada agente deve fazer na memória para ser possível a construção da experiência autobiográfica do robô. Os agentes, que estão foram desenvolvidos em um módulo separado no projeto [13], possuem mais casos de uso do que os mostrados pois foram descritos somente aqueles que deverão estar dentro do modelo de memória, que é o resultado desse projeto.

O Diagrama de Casos de Uso mostra que cada agente acessa o banco de dados, ou seja, instanciam sua própria conexão para a memória, já que estão em um arquitetura de multi-agentes, rodando em paralelo, se comunicando através de mensagens e podendo estar em locais físicos diferentes. E apesar do funcionamento em paralelo, existe um fluxo para guardar a informação, esse fluxo deve ser respeitado restringindo os receptores da mensagem que cada Agente envia, ou seja, determinado agente só pode receber a mensagem com a informação necessária para ser executado, se os agentes que realizam uma inserção prévia obrigatória terminaram de faze-la.

O *ImageAgent*, depois de obter uma imagem, terá de inseri-la na memória sensória, onde cada imagem será identificada por uma chave representada pelo *timestamp* de sua captura. Através dessa chave, que deverá ser passada para o *DetectionAgent*, este deve buscar na memória sensória a Imagem correspondente e guardar as coordenadas de onde as faces estão localizadas na imagem. A memória permite uma implementação futura onde se deseje salvar qualquer tipo de localização, por exemplo a de objetos, e não só de faces, já que o objeto *ObjectLocation* possui um atributo tipo, relacionado a um conceito da memória semântica que o caracteriza.

O *PersonRecognitionAgent*, através do mesmo *timestamp* poderá recuperar cada rosto encontrado buscando na memória sensória a imagem

original e a lista de localizações para realizar o reconhecimento. Tal atividade envolve, depois de extrair as características das faces, a busca na memória semântica por parâmetros de rostos conhecidos que apresentam maior grau de semelhança com a imagem obtida. Depois dessa consulta, caso seja reconhecido alguém, deve ser armazenado na memória episódica que determinado evento ocorreu ("Salvar Átimo Pessoa na Memória"), e associar que tal evento está relacionado tanto às imagens obtidas ("Adicionar relação Átimo - Sense") quanto à pessoa reconhecida ("Adicionar relação Conceito - Átimo").

Apesar de ambas as relações ("Adicionar relação Átimo - Sense"e "Adicionar relação Conceito - Átimo") ocorrerem conectando dois tipos de memória, foi escolhido mantê-las dentro da Memória Episódica, já que esta representa o centro da informação, sendo a duração do Átimo variável e dependente dos dados processados para serem relacionados na memória semântica e na memória sensória. Como explicado sobre o método *storeAtimo* na seção 5.3, que faz o controle de quando começa e termina um átimo, só será criado um novo se o método entender que isso ocorreu baseado na sua lógica interna, caso contrário, o átimo atual terá seu tempo final atualizado para o momento da última captura.

O *InstantEmotionRecognitionAgent*, depois de buscar na memória sensória as faces localizadas na imagem obtida, fará a identificação da emoção, buscando no memória semântica os parâmetros universais para cada uma das emoções e comparando com os que foram extraídos da face analisada. Caso o resultado seja determinado, o *InstantEmotionRecognitionAgent* irá criar outro Átimo na memória (respeitando a regra de controle de átimo citada no parágrafo anterior), com o mesmo momento de início (*timestamp*), porém um Átimo do tipo emoção, e não do tipo pessoa, também conectando-o ao conceito da emoção determinada que está na memória semântica, e a imagem localizada na memória sensória. Além disso, o agente deverá relacionar o Átimo emoção ao Átimo do tipo pessoa criado pelo *PersonRecognitionAgent*.

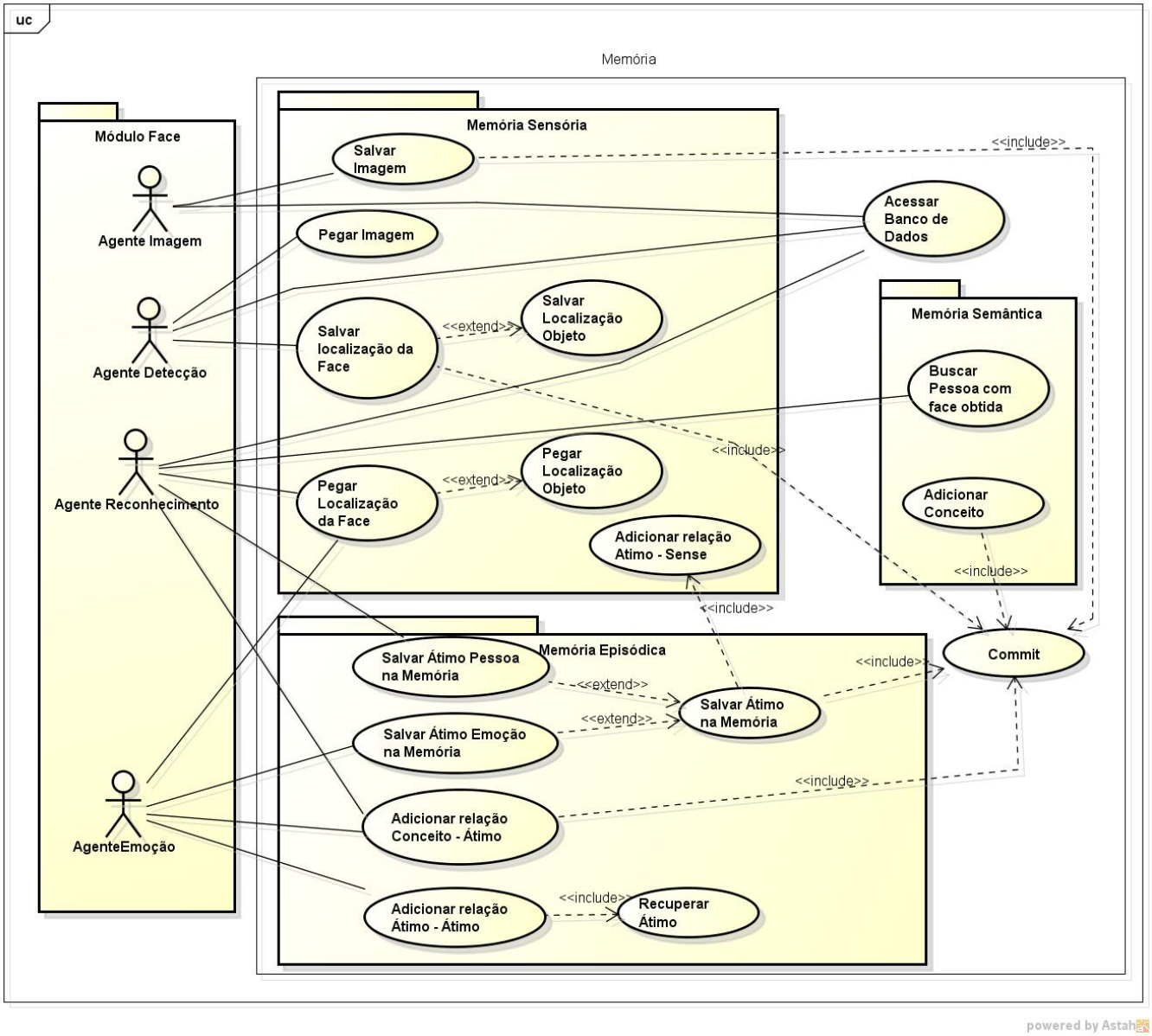

Figura 7 – Diagrama de Casos de Uso para o Modelo de Memória pelo Módulo Face

Alguns dos casos de uso mostrados, incluem o uso do *commit*, que confirma as transações feitas pelos métodos de armazenamento do modelo de memória, garantindo a integridade dos dados. Isso encerra o fluxo de informações que devem ser tratadas vindo do Módulo Face.

6.2 Demonstração das Informações Capturadas e Inseridas

Para executar o módulo face é necessário inicializar os agentes do JADE, para que depois disso cada Agente seja executado em paralelo esperando

mensagens de outros agentes chegarem para realizarem tarefas. O fluxo das informações descrito no capítulo acima, resultou nas informações que serão mostradas abaixo.
O programa foi executado com a pessoa "sabrina"em frente à câmera, tendo suas imagens capturas e seu rosto e suas emoções reconhecidas. O programa salvou no log às seguintes informações sobre o Agente de Reconhecimento de Pessoas

INFO: PersonRecognitionAgent :: in timestamp = 1414543071076 ==> found person = sabrina
INFO: PersonRecognitionAgent :: in timestamp = 1414543071755 ==> found person = sabrina
INFO: PersonRecognitionAgent :: in timestamp = 1414543072651 ==> found person =

Mostrando que foram capturadas duas imagens cujos *timestamps* estão indicados, e que para ambas as imagens, foi reconhecida a pessoa "sabrina". Abaixo podemos ver as emoções registradas:

INFO: InstantEmotionRecognitionAgent :: in timestamp = 1414543071076 ==> emotions recognized: Happiness
INFO: InstantEmotionRecognitionAgent :: in timestamp = 1414543071755 ==> emotions recognized: Happiness
INFO: InstantEmotionRecognitionAgent :: in timestamp = 1414543072651 ==> emotions recognized:

O terceiro registro de cada um dos *logs* está em branco pois depois de dois segundos a pessoa "sabrina"saiu da frente da câmera.
De acordo com as informações logadas é esperado que se tenha um átimo do tipo pessoa conectado ao conceito "sabrina", que já existia na memória semântica, pois foi uma das pessoas para as quais foram realizadas o

treinamento de reconhecimento. É esperado também que tenha sido criado um átimo do tipo emoção associado ao conceito *Happiness* (Felicidade), que já faz parte do modelo de ontologia existente. Ambos os átimos tem que ter seu inicio no *timestamp* 1414543071076, e seu fim, no momento 1414543072651 quando a "sabrina"não foi mais detectada.
Para verificar as informações salvas na memória sensória foi realizada a seguinte *query* na linguagem SPARQL, mostrada na figura 8 através do programa SQLDeveloper.

```
Select S, P, O From Table (Sem_Match(
'PREFIX id_s: <http://minerva.org/SensoryMemory/Sense/Id#>
SELECT ?s ?p ?o  WHERE {
{ id_s:1414543071076 ?p ?o . ?s ?p ?o} UNION
{ id_s:1414543071755 ?p ?o . ?s ?p ?o} UNION
{ id_s:1414543072651 ?p ?o . ?s ?p ?o}
}',
SEM_Models('SENSORYMEMORY'),null, null, null));
```

Figura 8 – *Query* para buscar *VisualSense* referente aos *timestamps* definidos acima

O resultado obtido está representado nas triplas mostradas na figura 9
Já a figura 10 mostra o resultado obtido, para somente um dos *Senses*, estruturado em grafos, o qual contém, como definido pela Ontologia na seção 5.2, as propriedades *startTime*, *endTime*, *has_File* que no caso é uma imagem e *has_faceLocation* um subtipo de *has_ObjetctLocation*.
Para a memória episódica, foi executada a *query* mostrada na figura 11 para se obter as triplas do banco de dados. As URIs buscadas como sujeitos para as triplas foram <http://minerva.org/EpisodicMemory/Atimo/Id#P1414543071076> e <http://minerva.org/EpisodicMemory/Atimo/Id#E1414543071076>.
A figura 12 mostra as informações dos átimos de emoção e de pessoas salvos no modelo de memória episódica, que também estão de acordo com a Ontologia pré-definida, enquanto que a 13 mostra esses mesmos dados dispostos numa estrutura de grafos.
Os resultados apresentados pelo sistema de memória se mostraram dentro

do esperado, sendo capaz de salvar as informações mantendo as regras da Ontologia definidas e realizando o controle dos átimos corretamente.

	S	P	O
1	http://minerva.org/SensoryMemory/Sense/Id#1414543071076	http://minerva.org/SensoryMemory/Property/has_startTime	1414543071076
2	http://minerva.org/SensoryMemory/Sense/Id#1414543071076	http://minerva.org/SensoryMemory/Property/has_endTime	1414543071076
3	http://minerva.org/SensoryMemory/Sense/Id#1414543071076	http://minerva.org/SensoryMemory/Property/has_faceLocation	rO0ABXNyABpmYWN1ZGV0ZWN0aW9uLmR1dGV
4	http://minerva.org/SensoryMemory/Sense/Id#1414543071076	http://minerva.org/SensoryMemory/Property/has_File	ORALL924
5	http://minerva.org/SensoryMemory/Sense/Id#1414543071755	http://minerva.org/SensoryMemory/Property/has_endTime	1414543071755
6	http://minerva.org/SensoryMemory/Sense/Id#1414543071755	http://minerva.org/SensoryMemory/Property/has_startTime	1414543071755
7	http://minerva.org/SensoryMemory/Sense/Id#1414543071755	http://minerva.org/SensoryMemory/Property/has_File	ORALL925
8	http://minerva.org/SensoryMemory/Sense/Id#1414543071755	http://minerva.org/SensoryMemory/Property/has_faceLocation	rO0ABXNyABpmYWN1ZGV0ZWN0aW9uLmR1dGV
9	http://minerva.org/SensoryMemory/Sense/Id#1414543072651	http://minerva.org/SensoryMemory/Property/has_endTime	1414543072651
10	http://minerva.org/SensoryMemory/Sense/Id#1414543072651	http://minerva.org/SensoryMemory/Property/has_faceLocation	rO0ABXNyABpmYWN1ZGV0ZWN0aW9uLmR1dGV
11	http://minerva.org/SensoryMemory/Sense/Id#1414543072651	http://minerva.org/SensoryMemory/Property/has_startTime	1414543072651
12	http://minerva.org/SensoryMemory/Sense/Id#1414543072651	http://minerva.org/SensoryMemory/Property/has_File	ORALL926

Figura 9 – Triplas contendo a informação salva na memória sensória

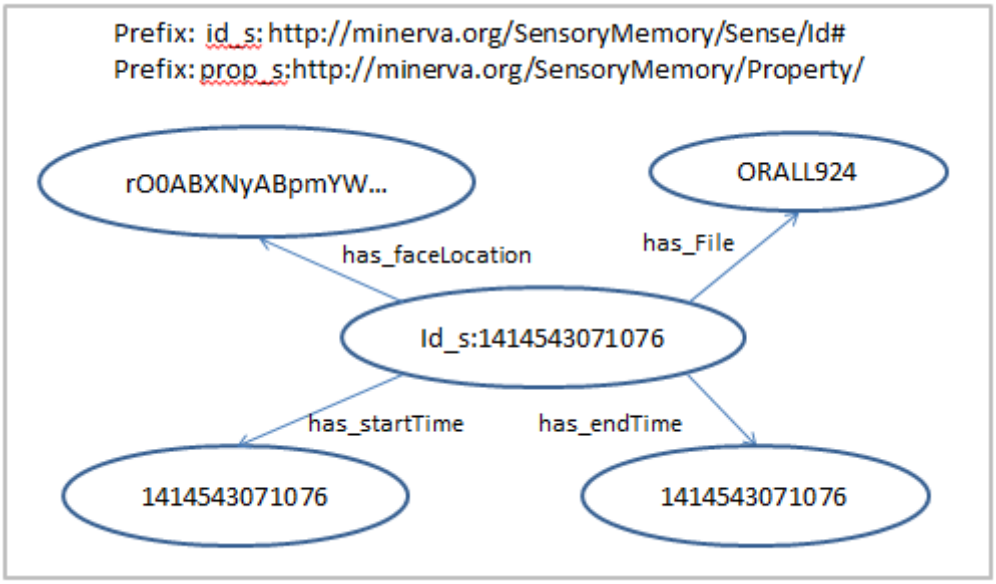

Figura 10 – Grafos representando as triplas inseridas no banco de dados

```
select p, o FROM TABLE (SEM_MATCH(
 'PREFIX i: <http://minerva.org/EpisodicMemory/Atimo/Id#>
 PREFIX prop_e: <http://minerva.org/EpisodicMemory/Property/>
 SELECT ?p ?o WHERE  { {i:P1414543071076 ?p ?o}
 UNION {i:E1414543071076 ?p ?o} }',
 Sem_Models('EPISODICMEMORY'),Null, Null, Null));
```

Figura 11 – *Query* para buscar os Atimos referente aos *timestamps* definidos acima

	P	O
1	http://minerva.org/EpisodicMemory/Property/has_startTime	1414543071076
2	http://minerva.org/EpisodicMemory/Property/has_GeneralChannel	http://minerva.org/SemanticMemory/GeneralOntology/Person
3	http://minerva.org/EpisodicMemory/Property/has_RelatedSense	http://minerva.org/SensoryMemory/Sense/Id#1414543071076
4	http://minerva.org/EpisodicMemory/Property/has_RelatedConcept	http://minerva.org/SemanticMemory/GeneralOntology/sabrina
5	http://minerva.org/EpisodicMemory/Property/has_InstantaneousEmotion	http://minerva.org/EpisodicMemory/Atimo/Id#E1414543071076
6	http://minerva.org/EpisodicMemory/Property/has_endTime	1414543071755
7	http://minerva.org/EpisodicMemory/Property/has_startTime	1414543071076
8	http://minerva.org/EpisodicMemory/Property/has_RelatedSense	http://minerva.org/SensoryMemory/Sense/Id#1414543071076
9	http://minerva.org/EpisodicMemory/Property/has_GeneralChannel	http://minerva.org/SemanticMemory/GeneralOntology/Emotion
10	http://minerva.org/EpisodicMemory/Property/has_RelatedConcept	http://minerva.org/SemanticMemory/GeneralOntology/Happiness
11	http://minerva.org/EpisodicMemory/Property/has_endTime	1414543071755

Figura 12 – Triplas inseridas no modelo da memória episódica

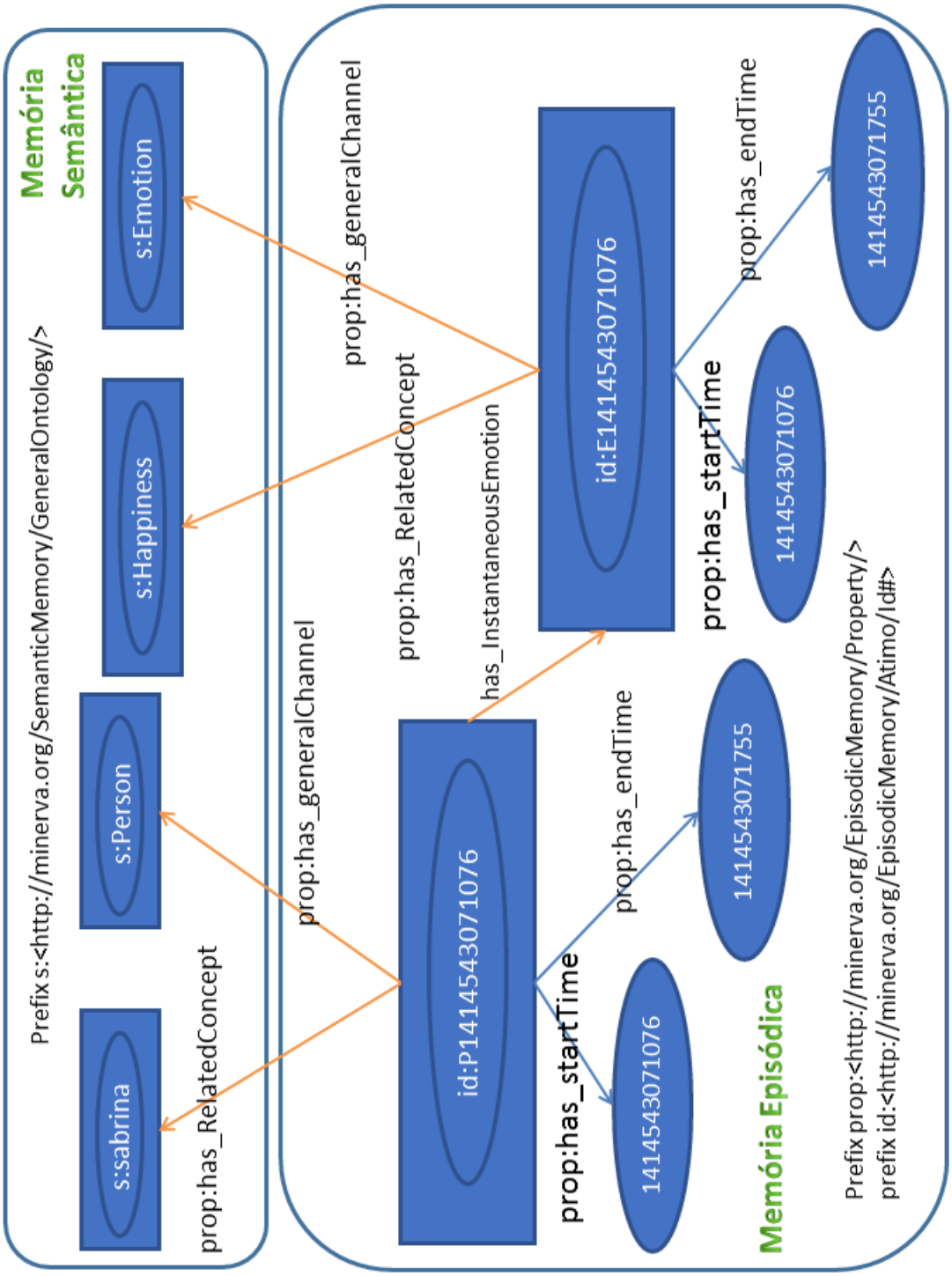

Figura 13 – Grafos representando as triplas inseridas no modelo da memória episódica

7 Comentários Finais

O trabalho apresentado aqui faz parte de um projeto de um robô sociável que visa a implementação de diversos módulos para que seja realizada toda a interação social do robô com pessoas. Desse modo o que foi desenvolvido representa uma pequena parte do que é necessário ser feito para que o robô seja capaz de dialogar com seres humanos utilizando informações recolhidas de encontros passados.

A implementação de memória teve como foco seu uso em diálogos, porém devido a complexidade do programa e dos testes que são necessários realizar, foi alcançada somente a implementação para o Módulo Face citado ao longo do capítulo anterior. Além disso, foi realizada apenas uma integração simples para testar a memória aqui desenvolvida, assim nem todas as funcionalidades existentes no Módulo Face foram integradas. Para que isso ocorra é necessário que o código desse módulo seja reescrito para utilizar as mesmas classes e sempre salvar e ler informações direto do banco de dados, e não salvando arquivos em pastas do sistema operacional.

Como melhorias, pode também ser necessário revisar o controle dos átimos, para que a coerência dos dados seja mantida mesmo se o robô interagir em um ambiente onde os usuários seriam hostis.

Além disso, é necessário que sejam criadas funções mais flexíveis para inserir dados na memória, o objetivo é encapsular alguns métodos que permitam a execução de *queries* em SPARQL, permitindo sua total implementação e completo uso de suas funcionalidades, evitando restringi-las.

Para a integração com o Gerenciador de Diálogos, será necessário criar classes, que estenderão a *Sense* para receber e tratar esse tipo de *input* sensório, tanto sonoro como textual. Essa integração apresenta uma complexidade maior que a do Módulo Face por necessitar de várias funcionalidades para que as falas sejam transformadas em triplas, para

serem salvas no banco de dados respeitando a linguagem OWL.
Atualmente o Módulo Face realiza o reconhecimento de rostos baseado em treinamentos que são realizados previamente, porém para que o robô possua um desempenho similar ao de seres humanos em interações é necessário que ele seja capaz de reconhecer e distinguir diferentes pessoas, mesmo que essas sejam desconhecidas, sem um treinamento prévio, o que é uma tarefa que ainda não é realizada com perfeição pelos sistemas disponíveis nessa área.
O modelo de memória desenvolvido nesse trabalho ainda necessita de muitas melhoras e adaptações para desempenhar junto com outros módulos tarefas que se aproximem das desejadas para um robô sociável, porém a estrutura aqui utilizada está baseada num banco de dados semântico com extensa documentação e sob o suporte da Oracle, uma empresa de grande porte na área de tecnologia preocupada com o crescente volume de dados que as aplicações necessitam, dessa forma acredita-se que o robô obtenha milhares ou milhões de registros diariamente, a tecnologia na qual está baseada irá acompanhar esse crescimento, de modo a conseguir manter a performance.
Além disso, as soluções de processamento e armazenamento em nuvem (*Cloud Computing*) que estão se tornando cada vez mais frequentes, apresentam-se como uma maneira adequada de se lidar com grande número de registros que o robô necessite para funcionar como desejado, já que os *hardwares* para tratar isso podem vir a ocupar muito mais espaço do que um robô teria para carregar toda essa capacidade de processamento e armazenamento, sendo necessário somente uma conexão confiável à Internet.
Outra vantagem que a solução da nuvem apresenta, é a possibilidade do compartilhamento de determinados conceitos, ou ontologias inteiras, entre diversos robôs, evitando retrabalho de equipes de pesquisadores.
A solução proposta nesse trabalho, que utiliza a linguagem OWL, da Web Semântica, tem como objetivo transformar todo o conhecimento humano a cerca do universo em um formato que as máquinas sejam capazes de

processar. Quando conseguirmos transformar as inúmeras páginas de conteúdo da Internet nesse formato, e criar métodos para semantizar falas, que automaticamente convertam sentenças inteiras em triplas, os robôs sociáveis sejam capazes de simular várias tarefas que só os humanos fazem atualmente, e para muitas delas, ter um desempenho muito superior.

Referências

[1] G. H. Silva. Construção de agentes inteligentes para a web semântica. [Online]. Available: https://linux.ime.usp.br/~cef/mac499-04/monografias/ghsilva/

[2] E. Tulving, "EPISODIC MEMORY : From Mind to Brain."

[3] M. a. Conway, "Episodic memories." *Neuropsychologia*, vol. 47, no. 11, pp. 2305–13, Sep. 2009.

[4] S. Lemaignan, R. Ros, L. Mosenlechner, R. Alami, and M. Beetz, "ORO, a knowledge management platform for cognitive architectures in robotics," *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 3548–3553, Oct. 2010. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5649547

[5] K. Addakiri and M. Bahaj, "Applying knowledge model to multi-agent system," *2013 International Conference on Computer Applications Technology (ICCAT)*, pp. 1–6, Jan. 2013.

[6] W. C. Ho, K. Dautenhahn, M. Y. Lim, P. a. Vargas, R. Aylett, and S. Enz, "An initial memory model for virtual and robot companions supporting migration and long-term interaction," *RO-MAN 2009 - The 18th IEEE International Symposium on Robot and Human Interactive Communication*, pp. 277–284, Sep. 2009. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5326204

[7] C. Roncancio, J. L. Rodriguez, E. Zalama, and J. Gomez G-B, "Improvement in service robot's interaction through case based reasoning," *The 2010 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–7, Jul. 2010. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5596619

[8] J. J. C. Graham Klyne. Resource description framework (rdf): Concepts and abstract syntax. [Online]. Available: http://www.w3.org/TR/2004/REC-rdf-concepts-20040210/

[9] Uma introdução a rdf e à api rdf de jena. [Online]. Available: http://jena.apache.org/tutorials/rdf_api_pt.html

[10] O. W. Group. Web ontology language (owl). [Online]. Available: http://www.w3.org/2001/sw/wiki/OWL

[11] D. L. M. Michael K. Smith, Chris Welty. Owl web ontology language guide. [Online]. Available: http: //www.w3.org/TR/2004/REC-owl-guide-20040210/#OwlVarieties

[12] (1999) Java agent development framework. [Online]. Available: http://jade.tilab.com/

[13] B. Tinen, "Sistema de identificação de emoções faciais com operação ao vivo," São Paulo, 2014.

[14] D. A. Alfenas and R. S. Paixão, "Sobre o uso de formalismos adaptativos no gerenciamento de diálogos em robôs sociáveis."