Universidade de São Paulo
Instituto de Matemática e Estatística
Departamento de Ciência da Computação

Monografia de Conclusão de Curso

Um Simulador de Comportamento Animal Baseado numa Especificação Formal da Análise do Comportamento de B. F. Skinner

19 de fevereiro de 2006

Paulo Salem
Autor

Ana Cristina Vieira de Melo
Orientadora

– É isto – disse Tarou. Por que o senhor mesmo demonstra tanta dedicação, já que não acredita em Deus? Sua resposta talvez me ajude a responder.

Sem sair da sombra, o médico disse que já respondera e que, se acreditasse num Deus todo-poderoso, deixaria de curar os homens, deixando a Ele esse cuidado. Mas que ninguém no mundo, não, nem mesmo Paneloux, que julgava acreditar, acreditava num Deus desse gênero, já que ninguém se entregava totalmente, e que nisso ao menos ele, Rieux, julgava estar no caminho da verdade, lutando contra a criação tal como ela era.

Albert Camus, *A Peste* [Cam03]

Sumário

1 Introdução

Este texto descreve o projeto de formatura de Paulo Salem, realizado sob orientação da professora Ana Cristina Vieira de Melo. O documento divide-se em duas partes, a saber, uma que trata de aspectos técnicos e outra que aborda considerações pessoais.

Na parte técnica começamos por prover uma breve introdução às áreas das quais nos valemos (Seção 2), procurando fornecer referências bibliográficas de modo a permitir que leitores interessados possam se aprofundar mais. Em seguida, mencionamos a metodologia adotada e as principais atividades realizadas (Seção 3). Por fim, apresentamos o que produzimos, explorando detalhadamente um breve trecho do trabalho (Seção 4), e elaboramos algumas conclusões gerais (Seção 5).

Nas considerações pessoais (Seção 6), buscamos relacionar o projeto com o curso de Bacharelado em Ciência da Computação (BCC). Assim, mostramos quais disciplinas nos parecem mais relevantes e como as outras atividades acadêmicas propiciadas pela Universidade de São Paulo contribuiram. Também fazemos uma avaliação geral do curso e da formação acadêmica que obtivémos.

Ao final do documento, provemos dois apêndices. O primeiro (Apêndice A) mostra o estado atual da especificação formal em Z, enquanto o segundo (Apêndice B) apresenta parte de uma ontologia que desenvolvemos no início do projeto.

O restante desta seção fornece uma visão geral do projeto.

1.1 Visão Geral

Desenvolvida por Burrhus Frederic Skinner, a Análise do Comportamento (em inglês, *Behavior Analysis*) é um ramo da Psicologia alinhado com os preceitos da chamada escola Comportamental (em inglês, *Behaviorism*). A Análise do Comportamento busca compreender o comportamento dos organismos através de relações entre os estímulos ambientais por eles recebidos e suas ações correspondentes. Tais relações estabelecem classes de comportamentos, as quais possuem propriedades específicas e bem definidas.

Do ponto de vista da Ciência da Computação, a simplicidade e precisão dessa teoria sugerem a possibilidade de uma implementação. Isto é, parece factível simular organismos em ambientes segundo as definições e resultados experimentais da Análise do Comportamento.

Um simulador assim concebido teria uma dupla utilidade. Em primeiro lugar, poderia servir como uma ferramenta para a Psicologia, na medida em que permitiria não só a realização de experimentos com organismos virtuais, como também a possibilidade de se testar novas teorias mediante o ajuste de parâmetros do simulador. Em segundo lugar, tal programa poderia ser adaptado não para simular criaturas, mas para dotar sistemas computacionais de capacidades comportamentais análogas.

No trabalho realizado, propômos a criação de tal simulador. Para tanto, o trabalho foi dividido em duas etapas fundamentais:

1. A especificação formal dos conceitos da Análise do Comportamento;
2. A construção de um programa que implemente tal especificação.

Embora os conceitos da Análise do Comportamento sejam razoavelmente claros e bem definidos, desconhecemos alguma formalização matemática deles. Por conseguinte, é necessário que estabeleçamos tal formalização antes de procedermos com a implementação. De outro modo, estaríamos desenvolvendo o programa com base em informações potencialmente incompletas e ambígüas, o que, claramente, seria indesejável.

No presente momento, ainda estamos trabalhando nessa formalização e é ela que apresentamos no restante desta monografia.

1.2 Aplicações do Simulador

Vemos o simulador como um *componente de software* que por si só não fornecerá serviços a usuários. Ele proverá um motor fundamental sobre o qual aplicativos poderão ser construídos. Tais aplicativos definirão o que consideram um organismo e seu ambiente, cabendo ao simulador explorar as conseqüências de tais definições. Embora poder-se-á definir organismos 'usuais' (i.e., animais), não será preciso limitar-se a isso, na medida em que qualquer definição que forneça sensores, atuadores e outros mecanismos poderá valer-se das capacidades do simulador. Por exemplo, podemos imaginar um processador de textos, digamos, que se adaptaria aos gostos do usuário, o qual poderia 'treiná-lo' tal como treinaria um cão.

Há muitas outras possibilidades de aplicações para o simulador. Embora estas não sejam o foco atual do projeto, podemos imaginar os seguintes tipos:

- Ferramenta de pesquisa em Psicologia;
- Ferramenta educacional em Psicologia;
- Ferramenta terapêutica;
- Jogos de computador;
- Componente de aprendizado para sistemas que necessitem de aprendizado computacional.

2 Conceitos e Tecnologias Estudadas

Nesta seção introduzimos os conceitos fundamentais das principais disciplinas presentes no projeto. Nosso objetivo, aqui, é dar ao leitor uma visão geral de cada área, sem explorar detalhes, mas provendo referências através das quais interessados possam aprofundar-se mais.

Começamos fazendo algumas ponderações a respeito da Psicologia em geral (Seção 2.1) para, em seguida, introduzir os principais elementos da teoria psicológica estudada, a Análise do Comportamento(Seção 2.1.1). Prosseguimos,

então, apresentando a área de Métodos Formais da Ciência da Computação (Seção 2.2), bem como a vertente particular utilizada no projeto, o método Z (Seção 2.2.1).

2.1 Psicologia

Nas ciências exatas há uma certa tendência à unanimidade com relação aos arcabouços teóricos utilizados. Na Matemática, por exemplo, os conceitos e métodos do Cálculo são únicos e amplamente aceitos – não há propostas realmente sérias para substituir os conceitos de limite, derivada e integral.

Já a Psicologia, bem como a maior parte das disciplinas ditas humanas, segue pelo caminho oposto. Não há uma 'teoria psicológica' central. Não há sequer uma definição única de Psicologia. A área é fragmentada em diversas escolas, cada uma propondo definições e métodos diferentes.

Reconhece-se, geralmente, o ano de 1879 como início da Psicologia enquanto ciência, devido à fundação do primeiro laboratório psicológico por Wilhelm Wundt. Wundt criara o método de *introspecção*, através do qual participantes submetidos a certas condições controladas relatavam suas experiências subjetivas. Desde então, muitas abordagens se seguiram, algumas prosseguindo na mesma linha, outras combatendo-na vigorosamente. Sucintamente, podemos listar algumas das escolas mais importantes que se desenvolveram ao longo do século XX:

- Estruturalismo;
- Funcionalismo;
- Psicanálise;
- *Gestalt*;
- *Behaviorism*;
- Humanismo;
- Cognitivismo.

Os leitores interessados podem obter uma introdução geral à Psicologia em [HVV03], onde obtivémos essas informações.

No nosso trabalho, adotamos as posições do *Behaviorism*[1], o qual apresentamos na seção seguinte.

[1] 'Behaviorism' é freqüentemente traduzido para o Português como 'Behaviorismo'. Essa tradução, contudo, nos parece péssima e preferimos ou manter o termo original, ou traduzí-lo como 'Comportamentalismo'.

2.1.1 *Behaviorism* e Análise do Comportamento

No início do século XX, a Psicologia ainda estava fortemente associada às idéias de Wundt e sua metodologia de introspecção. Alguns psicólogos, porém, acreditavam que tal metodologia era demasiadamente subjetiva e que, se a Psicologia visava um lugar na Ciência, ela deveria ser mais objetiva. Num famoso artigo [Wat13] de 1913, John B. Watson escreveu que:

> Psychology as the behaviorist views it is a purely objective experimental branch of natural science. Its theoretical goal is the prediction and control of behavior. Introspection forms no essential part of its methods (...)[2]

A filosofia por traz dessas idéias, o *Behaviorism*, pregava que o objeto de estudo da Psicologia não deveria ser a mente, uma vez que esta não é diretamente observável, mas sim o *comportamento* dos organismos, que é perfeitamente observável e mensurável. Assim, as teorias de cunho comportamental procuravam estabelecer relações diretas entre os estímulos ambientais e os comportamentos exibidos pelos organismos, sem passar pelo estudo intermediário da mente. A Figura 1 mostra um exemplo de experimento clássico dos primórdios desse paradigma.

Talvez o mais famoso representante do *Behaviorism* tenha sido Burrhus Frederic Skinner (Figura 2). Entre as décadas de 1930 e 1950, o professor Skinner desenvolveu sua própria vertente do *Behaviorism*, por ele denominada de Análise do Comportamento. Nosso projeto é baseado nessa teoria, sobretudo nos elementos descritos em [Ski53]. Examinemos então seus principais conceitos.

Em primeiro lugar, temos o conceito de *organismo*. Um organismo recebe *estímulos* do ambiente e produz *respostas comportamentais*. Tais respostas são divididas em duas classes, a saber, a dos *reflexos* e a dos *operantes*.

Reflexos são comportamentos muito simples, involuntários[3], inatos ao organismo e sem grandes possibilidades de mudança. São causados pela apresentação de certos estímulos, os quais levam o nome de *estímulos antecedentes*. Salivação e movimentos da pupila são exemplos de reflexos, cujos estímulos antecedentes são a apresentação de comida e de luz, respectivamente.

Os comportamentos operantes, por outro lado, são mais complexos, voluntários, aprendidos e com grandes possibilidades de mudanças. Operantes são definidos com base nas conseqüências que geram, isto é, na maneira pela qual operam no ambiente. Por exemplo, se pressionar um botão numa gaiola fornecer comida a um rato, o comportamento de apertar o botão passará a estar associado à provisão de comida, tornando-se, assim, um operante.

[2] A Psicologia como o comportamentalista a vê é um ramo puramente objetivo e experimental da ciência natural. Seu objetivo teórico é a predição e o controle do comportamento. A introspecção não pertence a nenhuma parte essencial de seus métodos (...)

[3] A rigor, todos os tipos de comportamentos são involuntários, na medida em que a Análise do Comportamento postula que o comportamento dos organismos é totalmente determinado pela estimulação ambiental. No entanto, utilizamos os termos 'involuntário' e 'voluntário' neste texto de forma não rigorosa, para fazer uso das noções intuitivas que o leitor provavelmente possui.

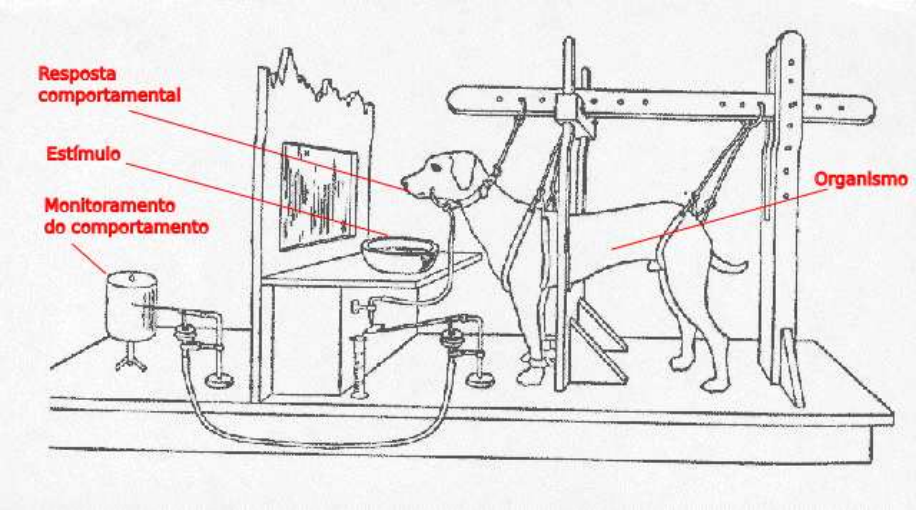

Figura 1: Um exemplo de experimento célebre. O organismo (cão) recebe estímulos do ambiente (comida) e, através de diversos processos, gera uma resposta comportamental (salivação) que pode ser monitorada.

Operantes podem ser *reforçados* ou *punidos*. Dizemos que é reforçado quando um estímulo agradável é alcançado pelo comportamento associado. Por outro lado, dizemos que é punido se o comportamento, que antes levava a um estímulo agradável, passar a levar a algum estímulo desagradável. É através dessas operações que criamos ou destruimos operantes. Quanto mais reforçado estiver um operante, mais provável o comportamento associado será. Deixado sem reforço ou punição, o operante tende a desaparecer, num processo conhecido como *extinção*.

A maneira pela qual os operantes são reforçados é conhecida como *escalonamento de reforço* e pode variar de diversas maneiras. Cada uma dessas variações possíveis produzirá um operante com propriedades específicas. Mediante escalonamentos de reforço distintos, podemos produzir operantes que demorem para ser extintos, mesmo sem reforços adicionais, ou operantes que durem pouco, mas produzindo uma alta freqüência de respostas.

Estímulos ambientais podem ser relacionados entre si, através de processos de *condicionamento*. Por exemplo, podemos condicionar, num cão, a provisão de comida ao som de um apito. Uma vez condicionado, o cão passará a salivar ao ouvir o apito, mesmo antes da comida ser apresentada (veja a Figura 1). Esse processo também é conhecido como *condicionamento clássico* e, em geral, é o experimento comportamental mais conhecido por leigos.

Um último ponto interessante da Análise do Comportamento é o tratamento dado às emoções. *Emoções* são definidas como alterações comportamentais. Então, por exemplo, dizemos que um organismo está *deprimido* se todos os

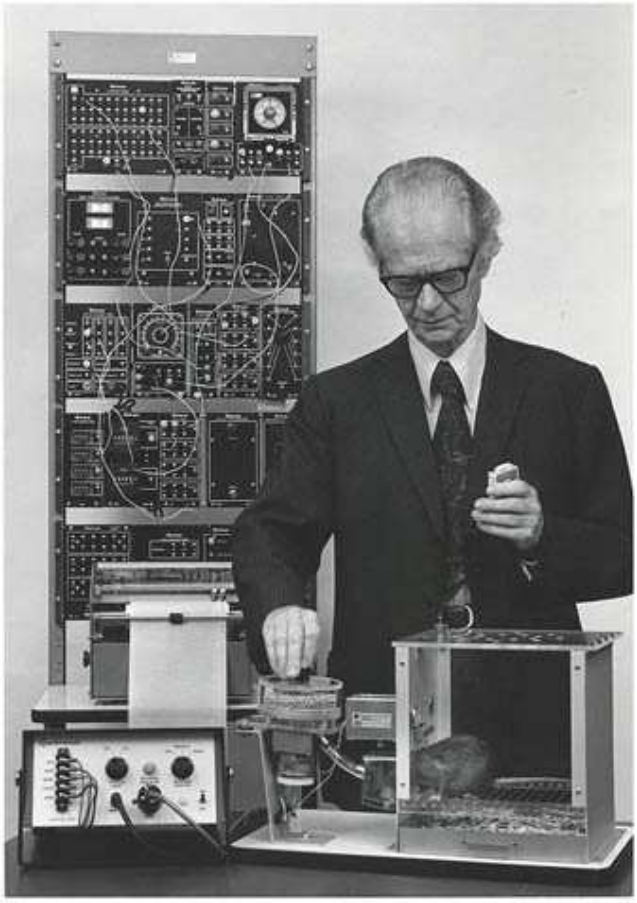

Figura 2: Professor Skinner realizando experimentos numa cobaia. A caixa em que o animal encontra-se é conhecida como 'skinner box', em virtude de ter sido um dos principais instrumentos propostos pelo professor. Ela é equipada com diversos aparatos que podem ser controlados pelo operador, incluindo luzes e grades elétricas.

seus comportamentos operantes tiveram suas taxas de emissão reduzidas em relação ao nível usual. Note que isso não nos diz nada a respeito da 'experiência subjetiva' do organismo – nosso enfoque é puramente comportamental.

Referências sobre Análise do Comportamento O livro mais conhecido nessa área provavelmente é o *Science and Human Behavior* [Ski53], do próprio professor Skinner. Ele contém tanto os aspectos técnicos da Análise do Comportamento, como também considerações filosóficos muito interessantes. Todavia, trata-se de uma obra antiga e, portanto, que não contempla muitos desenvolvimentos da área. Um texto atual e minucioso, de cunho mais técnico, pode ser encontrado em [Cat98].

2.2 Métodos formais

À primeira vista, a expressão *métodos formais* pode parecer vazia no contexto de desenvolvimento de software, na medida em que toda linguagem de programação é uma linguagem formal e, por conseguinte, pode ser vista como um 'método formal'. Contudo, não é esse o caso. A área de Métodos Formais abrange um nicho particular das linguagens formais, a saber, as linguagens de especificação. Essas linguagens têm como propósito permitir a definição de propriedades de interesse nos sistemas sendo construídos, da maneira mais abstrata possível. São baseadas em métodos e conceitos matemáticos, o que lhes confere semânticas bem definidas. Em geral, elas não visam descrever implementações completas, mas sim as restrições que devem ser impostas a quaisquer implementações. Assim, dizemos que métodos formais geram *especificações*, da mesma maneira que linguagens de programação convencionais geram programas executáveis.[4]

De modo geral, pode-se dizer que um método formal é caracterizado por ser:

Abstrato O método deve prover somente as abstrações necessárias para descrever-se as propriedades que se deseja modelar.

Formal Por 'formal', queremos dizer que as semânticas das especificações feitas com o método devem ser definidas de forma precisa.

Focado na especificação O método deve ser capaz de descrever características do sistema (*o quê* o sistema deve fazer) e não a maneira pela qual essas características devem ser implementadas (*como* o sistema deve implementar). Por exemplo, o método pode permitir a definição de invariantes presentes no sistema (i.e., relações entre variáveis do sistema), ou então a especificação de pré e pós condições de funções (i.e., relações entre as entradas e saídas das funções).

Voltado à verificação formal As especificações produzidas com o método devem ser passíveis de serem verificadas formalmente. Isto é, deve ser possível *demonstrar* teoremas a partir das especificações. Note que embora códigos fonte escritos em linguagens de programação convencionais

[4] No entanto, podemos ter especificações executáveis.

também possam ser verificados formalmente, essa prática raramente é adotada e, em geral, as próprias linguagens não são projetadas para esse fim.

Naturalmente, uma especificação formal de um programa precisa, em algum momento, ser transformada numa implementação. A transformação é alcançada através do processo de *refinamento*. Tal processo acrescenta detalhes à especificação, de forma a gerar ou uma especificação mais minuciosa, ou um código executável. O refinamento tem como característica fundamental o fato de que as propriedades da especificação original continuam valendo após o refinamento. Matematicamente, temos as seguintes implicações:

$$Propriedades_{novas} \Rightarrow Propriedades_{antigas}$$

$$Propriedades_{novas} \not\Leftarrow Propriedades_{antigas}$$

Métodos formais são particularmente valiosos para a construção de sistemas críticos (e.g., sistemas dos quais vidas humanas dependem), visto que a presença de certas propriedades de segurança pode ser *demonstrada*. Valendo-se de eventos desastrosos, como uma máquina de radioterapia que, inadvertidamente, provia doses letais de radiação aos pacientes, [Jac96a] argumenta em favor da abordagem formal no desenvolvimento de tais sistemas.[5]

Outra aplicação interessante para as metodologias formais é a descrição de domínios. Domínios assim descritos podem ser não só mais facilmente transformados em programas executáveis, como também mais facilmente compreendidos por seres humanos.

No tocante à compreensão humana, o enfoque abstrato é especialmente útil. Em [WD96], encontramos um exemplo interessante de como a eliminação de detalhes inúteis pode auxiliar usuários. O mapa original do metrô londrino (Figura 3) fora projetado em 1908 e era razoavelmente fiel aos detalhes geográficos. Em 1933, contudo, foi feita uma revisão do mapa, conhecida como *The Diagram*, com o propósito de eliminar detalhes irrelevantes para os usuários (Figura 4).

Embora esse exemplo pertença à área de *Design* Gráfico, ele evidencia o valor da abstração na descrição de um domínio.

Existem diversos métodos formais, os quais podem ser agrupados de acordo com o problema que procuram modelar, como, por exemplo:

- Sistemas seqüenciais;
- Sistemas concorrentes;
- Agentes móveis.

Além disso, os fundamentos matemáticos podem variar de acordo com com o estilo de especificação a ser adotada, que pode ser, por exemplo:

[5] A necessidade de métodos mais confiáveis, evidenciada pelo acidente descrito em [Jac96a], parece não ter sido percebida pela indústria. Em 2001, o mesmo tipo de acidente voltou a acontecer com certas máquinas de radioterapia da Multidata Systems, deixando ao menos oito mortos. [FDA03]

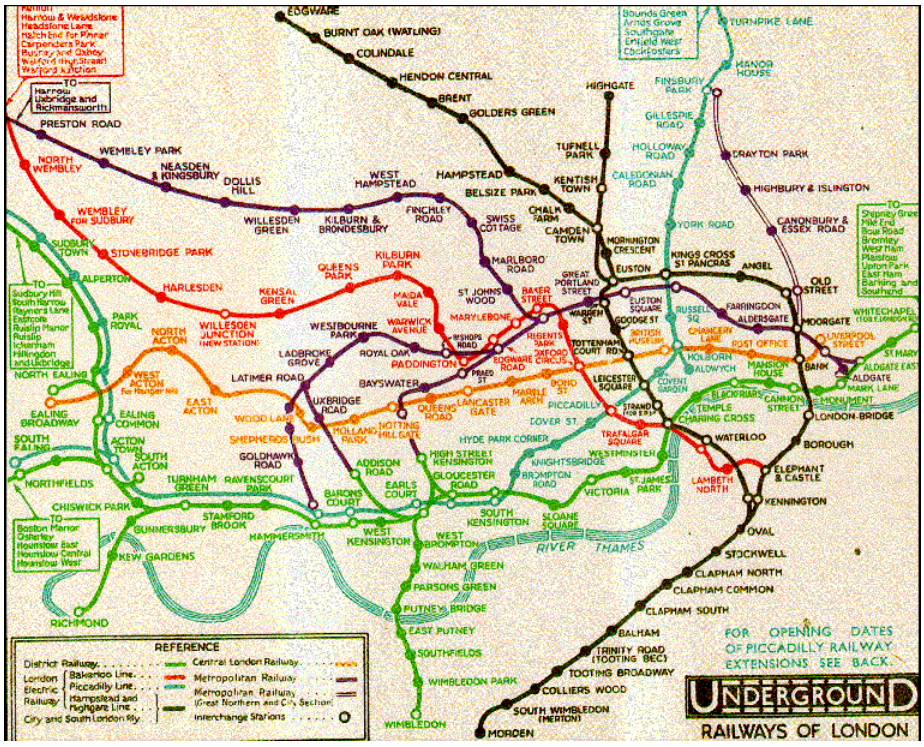

Figura 3: O mapa original do metrô londrino, fiel aos detalhes geográficos.

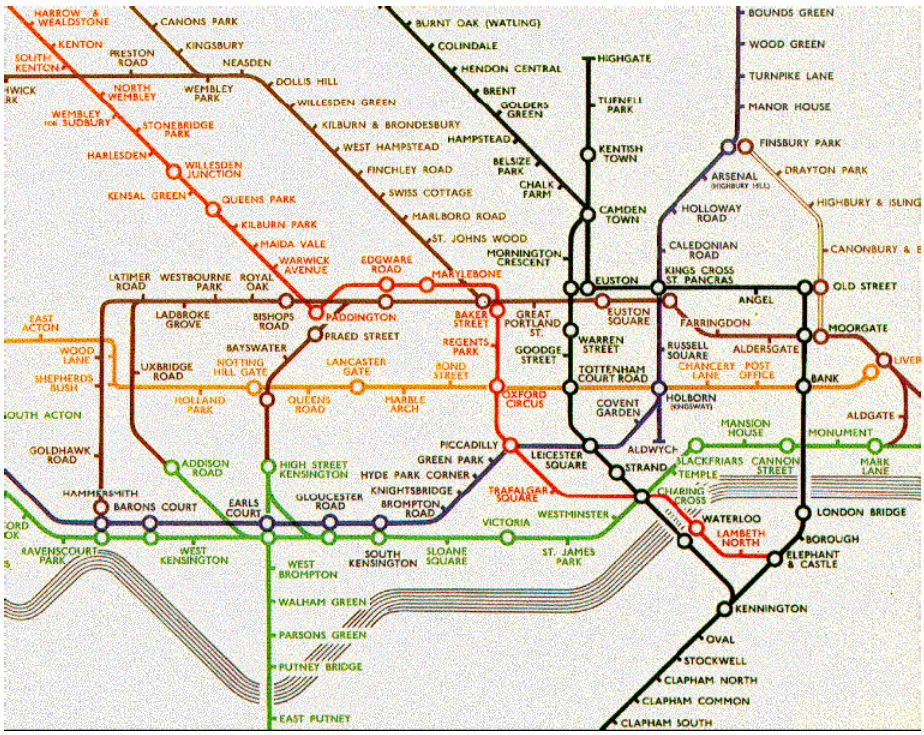

Figura 4: O novo mapa do metrô londrino, que ignora detalhes irrelevantes para os usuários.

- Algébrico;
- Baseado em modelos.

Como dito anteriormente, utilizamos o método formal Z para modelar um domínio, visando uma implementação. Na seção seguinte daremos uma breve introdução a esse método, de maneira que o leitor possa aprender seus conceitos fundamentais e conhecer sua bibliografia básica.

2.2.1 O Método Z

O método Z foi criado em fins da década de 1970 e desenvolvido durante a década de 1980 pela Universidade de Oxford e seus parceiros industriais. Entre outros sistemas, Z já foi usado para especificar máquinas de radioterapia[6] e sistemas gerenciadores de informação[7].

Segundo as categorias apresentadas na seção anterior, classificamos Z como um método formal para sistemas seqüenciais baseado em modelos. Programas,

[6] Veja [Jac96b], onde o autor descreve parte de sua experiência com essas máquinas.
[7] Veja o exemplo do sistema CICS em [WD96]

particularmente os escritos em linguagens imperativas, podem ser vistos como máquinas de estado. Cada valoração dada às variáveis do programa constitúi um estado e toda computação que leva a uma modificação dessa valoração define uma mudança de estado.

Com base nessa analogia, o método Z busca prover elementos para a descrição explícita e lógica dos estados e transições de sistemas computacionais. Por explícita, queremos dizer que a representação dessas abstrações fazem parte das primitivas da linguagem. Isto é, são conceitos reificados, e não meramente dedutíveis. Por lógica, nos referimos ao fato de que Z adota uma abordagem declarativa, baseada em Lógica de Primeira Ordem e Teoria dos Conjuntos. Ou seja, definimos, para cada estado e transição, aquilo que é *verdadeiro*. Desse modo, uma especificação em Z nada mais é do que a definição de todos os estados em que um sistema pode se encontrar.

Além de uma linguagem formal, Z também define uma notação gráfica, inspirada na notação matemática tradicional, com o propósito de facilitar a leitura de especificações.

Ao longo desta seção apresentaremos os conceitos fundamentais de Z, mediante um exemplo concreto.[8] Construiremos, passo a passo, um fragmento de uma especificação simplificada de uma bomba automática de insulina para diabéticos.[9] Comecemos examinando uma mesma sentença lógica escrita numa linguagem imperativa tradicional e em Z. Queremos afirmar que o nível de glicose no sangue do usuário está dentro de certos limites de segurança e que a bomba está desligada. Em C, por exemplo, temos algo como:

```
glucose_level >= MIN_GLUCOSE &&
glucose_level <= MAX_GLUCOSE &&
pump == OFF
```

Já em Z, a mesma informação pode ser escrita como:

$$Min_{glucose} \leq glucose\_level \leq Max_{glucose}$$
$$pump = off$$

A superioridade gráfica da notação de Z é clara. Ela traz a desvantagem, porém, de requerer mecanismos especiais para ser digitada, uma vez que teclados tradicionais não possuem teclas associadas a uma grande variedade de símbolos matemáticos. Para contornar esse problema, as especificações geralmente são escritas em LaTeX[Lam94], utilizando-se extensões específicas para Z, de modo que os símbolos especiais possam ser inseridos programaticamente. A especificação anterior, por exemplo, foi digitada da seguinte forma:

[8] O exemplo está em Inglês, pois adotamos a prática de escrever especificações exclusivamente nessa língua.

[9] Não exemplificamos o método com a especificação que desenvolvemos no projeto pois ela é demasiadamente complexa para encaixar-se em tão breve introdução. Ademais, métodos formais, tradicionalmente, têm grande aplicação em sistemas críticos, dentre os quais estão as bombas automáticas de insulina, visto que dosagens incorretas podem ser letais.

```
\begin{zed}
 Min_{glucose} \leq glucose\_level \leq Max_{glucose}
 \also
 pump = off
\end{zed}
```

Representação de Estados Em Z, estados são representados por uma abstração denominada *Schema*, a qual é composta por três seções:

Nome Um nome único para o *schema*;

Variáveis As variáveis que fazem parte do *schema*. Cada variável possui um tipo associado;

Invariantes Proposições lógicas que estabelecem o que é verdadeiro com relação às variáveis do *schema*. Z não adota nenhuma suposição de mundo fechado[10] : proposições que não são explicitadas não são necessariamente falsas. Essa propriedade é fundamental para o refinamento da especificação.

A notação gráfica associada a *schemas* é, talvez, a característica visual mais marcante do método. De relance, pode-se reconhecer que uma especificação está escrita em Z. A caixa do *schema* isola suas definições do resto do texto, o que facilita a leitura. Isso é particularmente importante pois não existe o conceito de *comentário* em Z. A notação Z é estritamente formal e cabe ao texto que a circunda explicá-la.

Continuando o exemplo, definiremos um *schema* para denotar o 'estado ideal' da bomba de insulina (i.e., o estado em que os níveis de glicose no sangue do paciente são adequados e no qual a bomba de insulina está desligada):

IdealState

$$Max_{glucose} : \mathbb{N}$$
$$Min_{glucose} : \mathbb{N}$$
$$pump : PUMP\_STATUS$$
$$glucose\_level : \mathbb{N}$$

$$Min_{glucose} \leq glucose\_level \leq Max_{glucose}$$
$$pump = off$$

O *nome* desse *schema* é *IdealState*. Suas variáveis são $Max_{glucose}$, $Min_{glucose}$, *pump* e *glucose_level*. Seus invariantes são as demais asserções. Observe que as variáveis possuem tipos associados. Três delas são números naturais (i.e., $\in \mathbb{N}$)

[10] A hipótese de mundo fechado nada mais é do que a suposição de que só é verdadeiro aquilo que se afirma explicitamente. Ou seja, que tudo que não for dito explicitamente é falso. [RN03]

e uma pertence a um conjunto chamado *PUMP_STATUS*. Em Z, podemos inventar novos tipos facilmente.

Schemas podem ser incluídos uns nos outros, através de importação. O resultado de uma importação nada mais é do que um novo *schema*, com as variáveis e invariantes tanto do que importa quanto do que é importado. Refaçamos o nosso *IdealState* de tal modo que os parâmetros de máximo e mínimo fiquem separados dos outras definições. Para tanto, criaremos um novo *schema* que definirá tais parâmetros:

SafetyParameters

$$Max_{glucose} : \mathbb{N}$$
$$Min_{glucose} : \mathbb{N}$$
$$Min_{glucose} \leq Max_{glucose}$$

Note que precisamos definir que o mínimo *realmente* é o mínimo. O invariante de *SafetyParameters* nos garante isso. Podemos, agora, reescrever *IdealState*:

IdealState$_2$

$$SafetyParameters$$
$$pump : PUMP\_STATUS$$
$$glucose\_level : \mathbb{N}$$
$$Min_{glucose} \leq glucose\_level \leq Max_{glucose}$$
$$pump = off$$

Semanticamente, *IdealState* e *IdealState*$_2$ são idênticos.

Representação de Transições Transições também são representadas utilizando-se *schemas*. Todavia, elas diferenciam-se da representação de estados mediante o uso de uma notação específica. Inspirando-se na notação tradicional da Física e da Matemática, Z convenciona o símbolo Δ (delta maiúsculo) como prefixo das variáveis sobre as quais deseja-se definir transições. Transições de estados também são chamadas de operações.

Modelemos, então, uma operação que atualiza o *schema IdealState*$_2$ de acordo com novas medições realizadas pelo equipamento.

IdealUpdateOp

$$\Delta IdealState_2$$
$$glucose\_reading? : \mathbb{N}$$
$$glucose\_level' = glucose\_reading?$$
$$pump' = pump$$
$$Min'_{glucose} = Min_{glucose}$$
$$Max'_{glucose} = Max_{glucose}$$

As variáveis seguidas por um '′' (apóstrofo) denotam o valor das variáveis *após* a aplicação da operação. As variáveis terminadas por um '?' (interrogação) denotam parâmetros de entrada da operação

As igualdades descritas nos invariantes acima são igualdades matemáticas, não atribuição de valores a variáveis. Assim, a ordem dos operandos é irrelevante. $glucose\_level' = glucose\_reading?$ equivale a $glucose\_reading? = glucose\_level'$. Observe também que o que *não muda* (e.g., o valor da variável *pump*) também deve ser definido. Se não dizemos nada a respeito de uma variável, estamos especificando que ela pode assumir qualquer valor após a operação – a hipótese de mundo fechado não vale em Z.

Os invariantes definem o que deve ser verdadeiro numa transição, mas não como essa verdade deve ser implementada. Além de especificar o que deve valer *após* a transição, também podemos definir o que deve ser verdade *antes* dela, as chamadas pré-condições. Para tanto, basta incluir invariantes que restrinjam as variáveis de entrada (i.e., parâmetros de entrada e valores do estado antes da aplicação da operação). Digamos, por exemplo, que desejamos estabelecer um intervalo mínimo de dez minutos entre as execuções da operação. Poderíamos modelar isso com uma variável e um invariante extras, da seguinte forma:

$IdealUpdateOp_2$

$\Delta IdealState_2$

$glucose\_reading? : \mathbb{N}$

$now? : TIME$

$now? \bmod 10 = 0$

$glucose\_level' = glucose\_reading?$

$pump' = pump$

$Min'_{glucose} = Min_{glucose}$

$Max'_{glucose} = Max_{glucose}$

Cálculo de *Schemas* Além do mecanismo de importação, existe outra maneira de se reaproveitar partes da especificação. *Schemas* também podem ser combinados logicamente de maneira a gerarem novos *Schemas*, através do chamado Cálculo de *Schema*. Esse recurso, quando bem utilizado, confere às especificações maior clareza, pois permite que sejam divididas de maneira a melhorar a compreensão do leitor.

Há também uma vantagem metodológica para o especificador. Alguns *schemas* podem ser úteis em pontos distintos da especificação e, portanto, convém definí-los isoladamente para posterior reuso. Isso não só reduz o trabalho do especificador, como também diminúi a possibilidade de erros, visto que não há duplicação da mesma informação para ser corrigida e atualizada.

Prosseguindo com nosso exemplo, vamos definir mais estados para o sistema, além do 'estado ideal'. Teremos algo como:

$$State == IdealState_2 \lor InsulinDeliveryState \lor DangerState$$

Esse trecho defini um novo *schema*, chamado *State*, como a disjunção de três outros *schemas*.

Um uso freqüênte e particularmente útil do Cálculo de *Schemas* é a definição das chamadas operações totais. Como vimos anteriormente, cada operação pode ter pré-condições. Dizemos que uma operação é total quando sua pré-condição é uma tautologia. Ou seja, quando ela prevê todos os possíveis casos da entrada. Nossa $IdealUpdateOp_2$ possui uma pré-condição que restringe sua execução a instantes múltiplos de 10. Assim, podemos definir uma outra operação, nula, que é executada nos demais instantes sobre o estado $IdealState_2$:

$$
\begin{array}{|l}
\hline
NullIdealUpdateOp_2 \\
\Delta IdealState_2 \\
now? : TIME \\
\hline
now? \bmod 10 \neq 0 \\
glucose\_level' = glucose\_level \\
pump' = pump \\
Min'_{glucose} = Min_{glucose} \\
Max'_{glucose} = Max_{glucose} \\
\hline
\end{array}
$$

Combinando ambas operações, temos uma nova operação que leva em conta todos os casos possíveis da entrada:

$$T\_IdealUpdateOp == IdealUpdateOp_2 \lor NullIdealUpdateOp_2$$

A pré-condição dessa nova transformação é:

$$(now? \bmod 10 = 0) \lor \neg\ (now? \bmod 10 = 0)$$

Isso sempre é verdadeiro e, portanto, a operação sempre pode ser executada.

Referências sobre Z Em [Jac96b] encontramos uma boa introdução, repleta de exemplos. O foco do livro está em mostrar como Z pode ser usado para modelar diversos tipos de sistemas computacionais. Já [WD96] fornece uma introdução mais formal ao método, dando grande ênfase a provas.

Atualmente, Z está definido num padrão ISO [ISO02]. Antes desse padrão, a referência mais completa para a linguagem era [Spi92].

Finalmente, diversos recursos relacionados ao método podems ser encontrados *online*, particularmente em [Bow].

2.2.2 Ontologias

Na Filosofia, Ontologia é um campo da Metafísica que estuda a existência *em si*. Esse uso clássico da palavra inspirou, na Ciência da Computação, a criação de métodos e técnicas para a representação de conhecimento. Assim, em

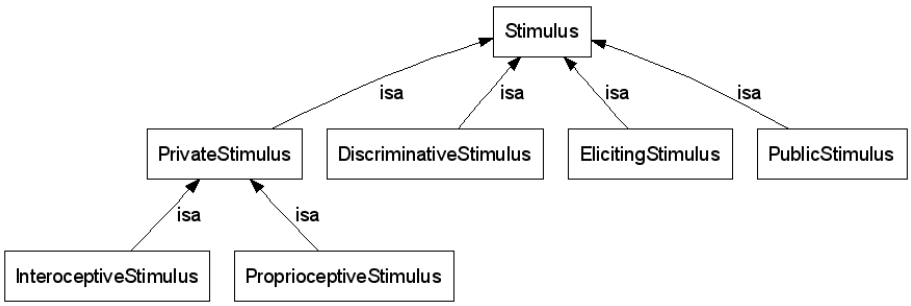

Figura 5: Um fragmento da nossa ontologia, descrevendo vários tipos de estímulo.

Computação, ontologias são especificações dos elementos existentes em domínios de aplicação, bem como de suas relações. Em geral, criam-se ontologias para permitir a comunicação de informações entre sistemas computacionais ou entre pessoas. Ou seja, ontologias podem ser vistas como linguagens para troca de informação.

Sistemas ou pessoas que adotam uma ontologia fazem o que se chama de *comprometimento ontológico*. Isto é, comprometem-se a reconhecer a semântica associada aos seus elementos.

O crescente interesse pela *Web* semântica [W3Ca] nos últimos anos acarretou em certa popularização do emprego de ontologias, o que, por sua vez impulsionou o desenvolvimento de padrões de descrição e ferramentas apropriadas. Nesse cenário, destaca-se a linguagem OWL [W3Cb] e a ferramenta Protégé [Pro].

Na fase inicial do nosso projeto, utilizamos essa ferramenta para construir um modelo ontológico dos principais conceitos da área da Psicologia que estudamos. Esse modelo teve como propósito melhorar nossa própria compreensão do assunto. Teve, portanto, um emprego educacional e não propriamente tecnológico, o que é um tanto incomum.[11]

O apêndice B contém parte da ontologia que produzimos para nosso estudo. Tomememos apenas um exemplo dela aqui, para visualizarmos em que constituise uma ontologia. A Figura 5 mostra um fragmento da ontologia que descreve várias classes de estímulo. A classe *Stimulus* é a mais geral possível e, a partir dela, estabelece-se uma hierarquia. Por exemplo, *InteroceptiveStimulus* é um tipo de *PrivateStimulus*, o qual, por sua vez, é um tipo de *Stimulus*.

Além de herança, há outras características que podem ser capturadas numa ontologia. Podemos, por exemplo, associar propriedades às classes e, mais ainda, relacionar as propriedades das classes entre si (e.g., podemos ter propriedades transitivas).

[11] Os conceitos capturados pela ontologia estão um tanto desatualizados com relação ao nosso conhecimento atual, visto que ela foi desenvolvida no início do projeto, principalmente para estudar sua viabilidade.

Referências sobre Ontologias Ontologias são uma parte da área de Representação de Conhecimento, à qual o capítulo 10 de [RN03] fornece uma introdução. No tocante à ontologias especificamente, o artigo [GG95] estuda os diversos empregos do termo e [Gru93] fornece uma definição[12] freqüentemente citada.

Atualmente, a tecnologia mais utilizada para descrever ontologias é a linguagem OWL [W3Cb]. A ferramenta de edição Protégé também é muito popular, principalmente para manipulação de OWL, e em seu *site* [Pro] pode-se encontrar diversos tutoriais.

3 Atividades Realizadas

A idéia para este projeto surgiu durante a leitura de *Science and Human Behavior*[Ski53]. Ao ler tal livro, o idealizador do projeto notou que os conceitos dessa teoria eram interessantes não só do ponto de vista Psicológico, mas também em seus aspectos computacionais, uma vez que fornecem elementos bem definidos para descrição de agentes.

Valendo-se de seu estudo prévio de ontologias, o autor esboçou uma ontologia que visava capturar os conceitos contidos no livro, de modo a estruturar sua própria compreensão do assunto. Conforme estudava, porém, notou que, embora aparentemente bem definidos, tais conceitos apresentavam imprecisões e que seria uma boa idéia formalizá-los com mais rigor, isto é, com algo mais poderoso do que uma descrição ontológica.

Foi nesse momento então que o autor teve a idéia de prosseguir seus estudos na área de Métodos Formais mediante a formalização matemática dos conceitos de Psicologia que estudara. Sendo sua orientadora especialista na área de Métodos Formais, ele propôs tal idéia como projeto de formatura, a qual a professora aceitou.

A partir desse momento, o autor passou a estudar o método Z, sobretudo a partir de [Jac96b], [WD96] e [ISO02]. Também buscou aprofundar o seu conhecimento em Psicologia mediante o estudo de [Cat98].

Quanto obteve um certo conhecimento mínimo de Z, o autor começou os esboços de sua formalização. Esta, porém, sofreu grandes mudanças ao longo do tempo, o que atrasou significativamente seu desenvolvimento. No presente momento ela ainda está em desenvolvimento.

4 Produtos obtidos

Como dito no início desta monografia, o objetivo final do projeto é implementar um simulador comportamental. Contudo, até o momento, ainda estamos terminando a especificação formal da teoria que visamos implementar. Como explicaremos na Seção 6, o trabalho *prático* com Z incorre em alguns problemas.

A formalização que estamos desenvolvendo é baseada na literatura do campo, sobretudo em [Ski53] e [Cat98]. Procuramos ser fiéis aos conceitos descritos na

[12] "an explicit and formal specification of a conceptualization"

literatura, na medida do possível. Contudo, no decorrer do processo, por vezes é necessário acrescentar ou remover conceitos, de maneira a possibilitar uma formalização apropriada[13].

No início do projeto, como apontado na Seção 3, desenvolvemos uma ontologia para a Análise do Comportamento, visando melhorar a nossa própria compreensão do assunto. Tal ontologia, portanto, foi um artifício educacional do qual nos valemos e não um objetivo *em si*. Assim, não a detalharemos aqui. Mas o apêndice B apresenta parte dela para os leitores interessados.

O apêndice A contém uma parte razoável da especificação formal *pura*[14] que elaboramos até o momento. Na presente seção, porém, apresentaremos apenas um fragmento dessa especificação, o qual define os processos diretamente relacionados a estímulos. Nosso objetivo, aqui, é explicar, mediante um exemplo detalhado, em que constitui a especificação que estamos desenvolvendo.

4.1 Exemplo: Especificação dos Processos de Estimulação

Uma função que leva estímulos ambientais em respostas comportamentais é a maneira mais abstrata possível de se modelar um organismo segundo os preceitos da escola Comportamental. Assim, vemos que o tratamento de estímulos é um componente fundamental para nossa modelagem. A especificação dos processos de estimulação visa definir os seguintes elementos:

- As entidades que são relevantes para o tratamento da estimulação. Isto é, os objetos que são submetidos a transformações e a partir dos quais outras estruturas são definidas;
- As relações relevantes entre as diversas entidades.;
- As funções que podem ser aplicadas a entidades. Aqui, nos referimos à acepção matemática de função;
- As operações que podem ser realizadas sobre entidades, relações e funções.

A seguir exploraremos cada uma dessas categorias.

4.1.1 Entidades

Estímulo Em primeiro lugar, precisamos afirmar que existe um conjunto de estímulos que o organismo é capaze que de receber e, em particular, que existe um estímulo neutro, isto é, que não afeta o organismo:

$$[Stimulus]$$

$$\mid \nu_s : Stimulus$$

[13] Por exemplo, criamos o conceito 'grafo de estímulos', como explicaremos adiante.

[14] Por 'pura' queremos dizer que não está acompanhada de comentários explicativos. Naturalmente, a especificação final, quando estiver pronta, terá que ser totalmente comentada e descrita num documento próprio.

Note que $[Stimulus]$ é um conjunto abstrato, isto é, não definimos exatamente quais são seus elementos. Isso nos dá a liberdade de refiná-lo posteriormente para conter os estímulos apropriados para as aplicações particulares.

Cada estímulo pode ser associado a um *status*, que determina se a o estímulo começou a ser recebido, terminou de ser recebido ou nenhuma dessas alternativas:

$$StimulusStatus ::= Beginning \mid Ending \mid None$$

$$\mid status_s : Stimulus \rightarrow StimulusStatus$$

Estimulação Com isso, podemos definir que um processo de estimulação é uma tripla, constituída por um estímulo, sua intensidade e seu *status*:

Stimulation

$$stimulus : Stimulus$$
$$intensity : Intensity$$
$$status : StimulusStatus$$

O processo de estimulação varia de organismo para organismo, de acordo com os seguintes parâmetros:

primaryStimuli Um conjunto de estímulos que são naturalmente agradáveis ou desagradáveis para o organismo;

$u_{primary}$ Uma função utilidade que define o grau de prazer ou dor causado pelos estímulos primários;

max_{delay} O máximo atraso entre dois estímulos, além do qual nenhum condicionamento pode ocorrer.

Assim, podemos definir o seguinte *schema*:

StimulationParameters

$$primaryStimuli : \mathbb{P}\, Stimulus$$
$$u_{primary} : \mathbb{P}\, Stimulus \rightarrow Intensity$$
$$max_{delay} : Duration$$

$$\mathrm{dom}\, u_{primary} = primaryStimuli$$

Observe que o domínio da função $u_{primary}$ está restrito, por um invariante, ao conjunto de estímulos primários. Isso é importante, pois adiante definiremos uma outra função utilidade, a qual cobrirá todos os estímulos e fará uso dessa que acabamos de definir.

Atenção Finalmente, o conceito de atenção é fundamental para um correto tratamento de estímulos. Essencialmente, a atenção é um conjunto de estímulos no qual o organismo está interessado num determinado instante. Ainda não detalhamos muito esse conceito. Mesmo assim, incluimos os *schemas* necessários:

AttentionParameters

$$\_\_\_\_\_$$

Attention

$$saliences : Stimulus$$

$$\_\_\_\_\_$$

4.1.2 Relações

O comportamento dos organismos depende muito da capacidade deles de aprender sobre como os estímulos ambientais estão relacionados. Às vezes, é útil considerar dois estímulos que, na verdade, são diferentes, como equivalentes. Por exemplo, se, através de procedimentos experimentais, nós pudermos fazer com que tanto a presença de uma luz vermelha, quanto a de uma luz verde acarretem nas mesmas conseqüências (e.g., comida), por quê uma cobaia faminta deveria distinguir entre essas cores?

Por outro lado, há ocasiões nas quais a relação apropriada é de implicação, não de equivalência. No exemplo anterior, as luzes são sempre seguidas por comida, porém a presença de comida não é, necessariamente, seguida pelas luzes. Isto é, o aprendizado leva em conta a ordem da apresentação dos estímulos.

A seguir examinaremos cada uma dessas relações.

Comecemos pela relação de implicação. Matematicamente, podemos dizer que é uma relação de ordem parcial (i.e., uma relação reflexiva e transitiva). A essa relação também associamos uma função que determina sua intensidade:

StimulusImplication

$$\_ \rightsquigarrow_s \_ : Stimulus \leftrightarrow Stimulus$$

$$intensity : Stimulus \times Stimulus \rightarrow Intensity$$

$$\_\_\_\_\_$$

$$\forall s_1, s_2 : Stimulus \bullet$$

$$\quad s_1 \rightsquigarrow_s s_2 \Leftrightarrow s_1 \rightsquigarrow s_2$$

$$\forall s_1, s_2 : Stimulus \mid s_1 \rightsquigarrow_s s_2 \bullet$$

$$\quad \exists i : Intensity \bullet (s_1 \rightsquigarrow_s s_2 \mapsto i) \in intensity$$

Note que essa relação de implicação pode ser vista como um grafo dirigido (Figura 6), no qual os vértices representam estímulos e as arestas indicam o condicionamento entre estímulos. Mais ainda, as arestas podem ter pesos, se a intensidade desse condicionamento deve ser levada em conta. Chamamos tal estrutura de *grafo de estímulos*.

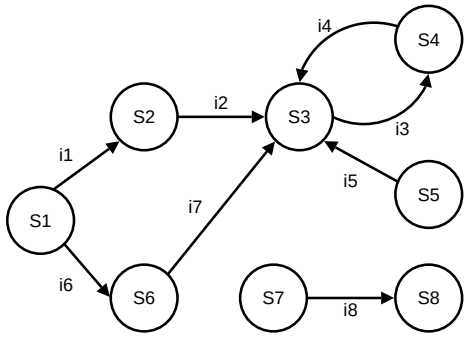

Figura 6: Um exemplo de relação de implicação representada por um grafo dirigido.

Valendo-nos da relação de implicação, podemos facilmente definir uma relação de equivalência simplesmente exigindo que se um estímulo s_1 implica um s_2, então s_2 também deve implicar s_1:

StimulusEquivalence

$$\_ \equiv_s \_ : Stimulus \leftrightarrow Stimulus$$
$$intensity : Stimulus \times Stimulus \rightarrow Intensity$$

$$\forall\, s_1, s_2 : Stimulus \bullet$$
$$s_1 \equiv_s s_2 \Leftrightarrow s_1 \rightsquigarrow_s s_2 \land s_2 \rightsquigarrow_s s_1$$
$$\forall\, s_1, s_2 : Stimulus \mid s_1 \equiv_s s_2 \bullet$$
$$\exists\, i : Intensity \bullet (s_1 \equiv_s s_2 \mapsto i) \in intensity$$

Novamente, podemos ver tal relação como um grafo. Porém, neste caso, as arestas não são dirigidas (Figura 7).

4.1.3 Funções

Tendo estabelecido as possíveis relações entre estímulos e a existência de alguns estímulos primários, podemos definir uma *utilidade* para qualquer estímulo dado. Existem, porém, muitas maneiras razoáveis de definirmos tal utilidade. Por esse motivo, procuramos dar essa definição em níveis mais abstratos primeiro e, em seguida, refiná-los. Dessa maneira, é possível especificar novos refinamentos.

No nível mais abstrato, definimos que o estímulo neutro, ν_s, possui utilidade nula e que a utilidade dos outros estímulos é uma função que depende do estado emocional e motivacional[15] do organismo.

[15] Traduzimos o termo 'drive' para 'motivação'.

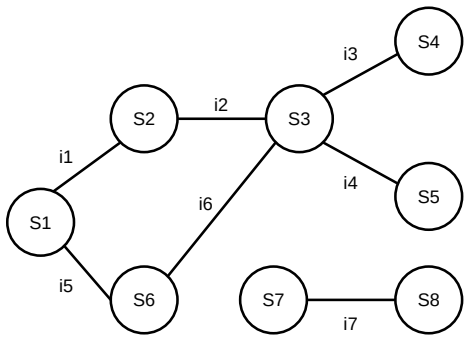

Figura 7: Um exemplo de relação de equivalência representada como um grafo não dirigido.

StimulusUtility

$$
\begin{array}{l}
StimulationParameters \\
EmotionSubsystem \\
DriveSubsystem \\
u_s : Stimulus \rightarrow Intensity \\
\hline
u_s(\nu_s) = 0 \\
\exists f : Stimulus \times \mathbb{P}\, Emotion \times \mathbb{P}\, Drive \rightarrow Intensity \bullet \\
\quad u_s(s) = f(s, activeEmotions, activeDrives)
\end{array}
$$

Note que esse *schema* utiliza dois outros ainda não definidos, o *EmotionSubsystem* e o *DriveSubsystem*. Esses *schemas* não pertencem à especificação dos processos de estimulação e, portanto, não estão incluídos no que vimos nesta seção até o momento.

Com essa definição básica, podemos proceder num refinamento. Vamos especificar que a utilidade de um estímulo arbitrário é igual à utilidade do maior estímulo alcançável no grafo de estímulos definido anteriormente. Para tanto, temos dois *schemas*:

StimulusUtilityBase$_1$

$$
\begin{array}{l}
StimulusUtility \\
StimulusImplication \\
base : Stimulus \rightarrow Intensity \\
\hline
\forall\, s : Stimulus \bullet \\
\quad (\exists\, s_p : primaryStimuli \bullet \\
\qquad base(s) = u_{primary}(s_p) \land \\
\qquad \forall\, s_q : primaryStimuli \mid s \rightsquigarrow_s s_q \bullet \\
\qquad\quad u_{primary}(s_p) \geq u_{primary}(s_q) \land \\
\qquad (s \rightsquigarrow_s s_p)) \lor \\
\quad (\forall\, s_p : primaryStimuli \bullet \\
\qquad s \not\rightsquigarrow_s s_p \land \\
\qquad u_s(s) = 0)
\end{array}
$$

StimulusUtility$_1$

$$
\begin{array}{l}
StimulusUtilityBase_1 \\
EmotionalRegulators \\
DriveRegulator \\
\hline
\forall\, s : Stimulus \bullet \\
\quad u_s(s) = driveRegulators(s, emotionalRegulators(s, base(s)))
\end{array}
$$

No primeiro, definimos a função *base*, que especifica a utilidade do estímulo primário alcançado. No segundo, efetivamente definimos a utilidade para qualquer estímulo, valendo-nos tanto da função *base*, quanto de funções definidas pelos subsistemas emocional e motivacional. A Figura 8 ilustra o grafo de estímulos levando em conta os estímulos primários.

É interessante notar que existem muitas outras questões em aberto no tocante ao cálculo da utilidade. Primeiramente, é claro, as definições dadas não fazem uso da intensidade da relação entre os estímulos. Embora complicado, esse refinamento pode ser feito.

Outras questões, porém, não são de fácil solução. Por exemplo, definimos uma busca no grafo de estímulos, mas resta especificar como essa busca é feita (e.g., em profundidade, em largura) e se isso é relevante para organismos reais. Como não ainda temos respostas razoáveis para essas perguntas, não especificamos o método de busca. Numa implementação, idealmente, o algoritimo de busca seria substituível, de modo a permitir experimentos com várias possibilidades.

4.1.4 Operações

Já definimos a estrutura das relações entre estímulos, mas ainda nos falta especificar de que modo essas relações são criadas. O processo de condicionamento, que definiremos a seguir, é responsável por isso.

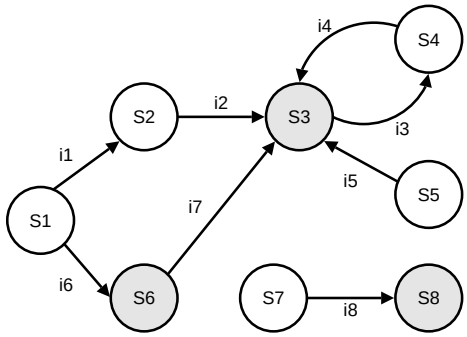

Figura 8: Um exemplo da aplicação da utilidade de estímulos considerando a relação de implicação de estímulos. Os vértices sombreados representam os estímulos primários. Com a definição de $StimulusUtility_1$, temos os seguintes resultados: (a) A utilidade de s_1 é a maior entre as de s_3 e s_6; (b) se s_3 tem uma utilidade maior do que a de s_6, então s_1, s_2, s_4 e s_5 têm todos a mesma utilidade.

Dados dois estímulos, o organismo pode aprender que eles ocorrem em uma certa ordem. Esse processo de aprendizado, chamado de condicionamento, é definido por duas operações. Uma é responsável por fortalecer esse elo entre estímulos, enquanto outra os enfraquece, promove um decaimento da associação:

ConditioningOp

$\Delta StimulusImplication$

$s_1? : Stimulus$

$s_2? : Stimulus$

$delay? : Duration$

$delay? \leq max_{delay}$

$s_1? \rightsquigarrow'_s s_2?$

$(s_1? \rightsquigarrow_s s_2?) \land intensity(s_1?, s_2?) < 1 \Rightarrow$
$\quad intensity'(s_1?, s_2?) > intensity(s_1?, s_2?)$

$(s_1? \rightsquigarrow_s s_2?) \land intensity(s_1?, s_2?) = 1 \Rightarrow$
$\quad intensity'(s_1?, s_2?) = intensity(s_1?, s_2?)$

$s_1? \not\rightsquigarrow_s s_2? \Rightarrow$
$\quad intensity'(s_1?, s_2?) > 0$

ConditioningDecayOp

$\Delta StimulusImplication$

$s_1? : Stimulus$

$s_2? : Stimulus$

$delay? : Duration$

$delay? > max_{delay}$

$s_1? \rightsquigarrow_s s_2?$

$intensity'(s_1?, s_2?) < intensity(s_1?, s_2?)$

$intensity'(s_1?, s_2?) \leq 0 \Rightarrow$
$\quad s_1? \not\rightsquigarrow'_s s_2?$

Com isso, podemos definir uma operação de condicionamento como a disjunção dessas duas outras operações:

$$T\_ConditioningOp_1 == ConditioningOp \lor ConditioningDecayOp$$

Todavia, essas definições não restringem o valor exato da mudança na intensidade da associação entre dois estímulos. Elas apenas dizem se a intensidade deve aumentar ou diminuir.

Como não sabemos como essa taxa de mudança é dada em organismos reais, não podemos fixá-la aqui. Porém, podemos definir refinamentos que especifiquem *possibilidades* a respeito dessa taxa. Assim, definimos um refinamento simples que adota uma taxa linear. Comecemos por um *schema* que armazena parâmetros importantes da operação:

LinearConditioningParameters

$StimulationParameters$

$\alpha_s : Intensity$

$increment : Intensity$

$newIntensity : Intensity$

$delay? : Duration$

$increment = \alpha_s / delay?$

LinearConditioningOp então, refina *ConditioningOp*:

LinearConditioningOp

$$StimulusConditioningOp$$
$$LinearConditioningParameters$$

$$newIntensity = intensity(s_1?, s_2?) + increment$$
$$intensity(s_1?, s_2?) + increment \leq max_{intensity} \Rightarrow intensity' = intensity \oplus (s_1?, s_2?, newIntensity)$$
$$intensity(s_1?, s_2?) + increment > max_{intensity} \Rightarrow intensity' = intensity \oplus (s_1?, s_2?, max_{intensity})$$

E *LinearConditioningDecayOp* refina *ConditioningDecayOp*:

LinearConditioningDecayOp

$$ConditioningDecayOp$$

$$newIntensity = intensity(s_1?, s_2?) - increment$$
$$intensity(s_1?, s_2?) - increment \geq 0 \Rightarrow intensity' = intensity \oplus (s_1?, s_2?, newIntensity)$$
$$intensity(s_1?, s_2?) - increment < 0 \Rightarrow intensity' = intensity \oplus (s_1?, s_2?, 0)$$

Finalmente, $T\_ConditioningOp_2$ refina $T\_ConditioningOp_1$:

$$T\_ConditioningOp_2 == LinearConditioningOp \lor LinearConditioningDecayOp$$

5 Conclusões

A especificação ontológica que produzimos logo no início do projeto mostrou a utilidade das ontologias enquanto ferramentas educacionais. Pelos nossos conhecimentos, esse não é um uso comum de ontologias. Ademais, acreditamos que essa ontologia possa ser melhorada a ponto de tornar-se um artefato tecnologicamente útil.

Durante a construção da especificação em Z, notamos diversos problemas práticos. Como já comentamos antes, não existem muitas ferramentas de apoio à edição. Isso foi particularmente problemático pois nossa especificação foi completamente alterada por mais de uma vez, o que acarretou em grandes atrasos. Num projeto de pesquisa, como o nosso, atrasos são aceitáveis. Porém, em aplicações industriais, que são os objetivos finais de quaisquer métodos de desenvolvimento, tais atrasos não são admissíveis. Pensamos, assim, que é importante que a comunidade de Métodos Formais dê mais atenção ao desenvolvimento de ferramentas de edição.

No tocante ao processo de formalização, percebemos que algumas formalizações são particularmente difíceis de serem feitas. Muitos dos conceitos encontrados na literatura psicológica são incompletos e, por vezes, ambígüos. Esse tipo de imprecisão pode ser aceitável para os especialistas humanos, capazes de

compreender as idéias principais dos textos e completá-las com conhecimentos e conjecturas próprias. Mas nós, que visamos uma implementação computacional, precisamos de definições exatas. Foi curioso estudar essa fronteira entre a precisão e a imprecisão. Acreditamos, baseando-nos nessa experiência, que há muito a se ganhar em formalizações matemáticas de áreas tradicionalmente não matemáticas. Dentre os ganhos que se pode obter, além dos aspectos computacionais, lembramos a facilitação da compreensão do assunto e o reaproveitamento de conhecimento já existente acerca das estruturas matemáticas usadas na formalização (e.g., se podemos modelar um problema como um grafo, então podemos utilizar a teoria de grafos para tratá-lo).

Finalmente, avaliamos que subestimamos a complexidade do projeto, ainda que ele continue nos parecendo viável.

6 Aspectos Subjetivos

Nesta seção tratarei[16] dos aspectos subjetivos relacionados tanto ao projeto de formatura quanto ao BCC de forma geral. Para fins de análise, convém dividir as minhas experiências nos seguintes grupos:

- Impressões gerais a respeito do BCC e do projeto (Seção 6.1);
- Desafios e frustrações (Seção 6.2);
- Disciplinas obrigatórias e optativas (Seção 6.3);
- Iniciação científica (Seção 6.4);
- Atividades acadêmicas extra-curriculares (Seção 6.5).
- Interação com pessoas (Seção 6.6);

Ao final, faço algumas considerações acerca de meus planos para o futuro (Seção 6.7).

6.1 Impressões Gerais

De modo geral, acredito que o BCC me foi muito útil e que meu projeto de formatura reflete razoavelmente bem a formação que adquiri.

Através do BCC, não só ganhei conhecimento, como também novas e mais corretas formas de pensar. Ao entrar no curso, minhas habilidades em Computação limitavam-se a programar em *Visual Basic* de modo desestruturado. Ao sair, sinto-me em condições, por exemplo, de programar um '*Visual Basic*' *em si*.

Porém, creio que a influência mais importante do curso não tenha sido nesse lado prático, de desenvolvimento de *software*, mas sim nos aspectos teóricos,

[16] Utilizarei a primeira pessoa do singular nesta seção, visto que trata-se de opiniões pessoais e não acadêmicas.

sobretudo no tocante à Matemática e à Lógica, áreas das quais eu nada sabia *realmente*. A relevância dessa educação teórica está no fato de que ela se estende para além da Computação, uma vez que fornece ferramentas mentais para se trabalhar eficientemente em qualquer área do conhecimento.

Assim, meu projeto de formatura é meramente uma conseqüência desse fato. Embora o currículo do BCC não tenha me fornecido, em momento algum, qualquer conhecimento de Psicologia, o *modo de pensar* que adquiri durante o curso me permitiu estudar cuidadosamente uma parte da literatura psicológica e enxergar nela diversas estruturas e possibilidades. Na Seção 6.3 abordarei detalhadamente o papel desempenhado por cada disciplina no meu projeto.

6.2 Desafios e Frustrações

Como é inevitável para a grande maioria dos alunos de graduação no IME, eu também tive dificuldades, a princípio, para acompanhar as disciplinas de cunho matemático. Contudo, ao contrário de muitos colegas, vejo isso como algo bom. Afinal, se superei muitas dessas dificuldades, significa que me tornei uma pessoa melhor e que o tempo gasto não foi em vão.

Com relação ao projeto, os desafios foram mais amenos. Em primeiro lugar, há o problema óbvio: aprender sobre uma área desconhecida, a Psicologia. Superei-o mediante a leitura de livros acadêmicos da área, processo este que me tomou muitos meses.

Instruir-me no método Z também não foi tarefa simples, visto que trata-se de um paradigma diferente do que se treina durante o BCC. Ademais, as ferramentas existentes para o método são bem limitadas, o que, por vezes, torna seu uso penoso e ineficiente. Por exemplo, não existem ferramentas de refatoramento automático, de modo que certas alterações que afetam toda a especificação precisam ser feitas manualmente, o que é um processo lento e propenso a erro. Ademais, a experiência mostrou que as primeiras especificações tendem a não serem satisfatórias, o que me obrigou, por mais de uma vez, a recomeçar a especificação dos mesmos elementos.

Finalmente, e como conseqüência desses fatos, minha maior frustração foi não ter implementado o simulador previsto pelo projeto. Isso será feito, contudo.

6.3 Disciplinas Relevantes

As disciplinas que me foram relevantes no projeto podem ser divididas em dois conjuntos, a saber, o das que tiveram influência direta e o das que exerceram influência indireta, secundária. Estão no primeiro conjunto Métodos Formais em Programação (MAC 239), Introdução à Inteligência Artificial (MAC 425), Conceitos Fundamentais de Linguagens de Programação (MAC 316), Algorítmos em Grafos (MAC 328), Álgebra II (MAT 213), Engenharia de Software (MAC 332), Tópicos de Programação Orientada a Objetos (MAC 413), Laboratório de Programação I e II (MAC 211 e MAC 242, respectivamente). Já no segundo grupo coloco Introdução à Computação Gráfica (MAC 420), Programação Linear (MAC 315), Introdução aos Fundamentos da Matemática (MAT

350) e diversas outras disciplinas que são analisadas conjuntamente. A seguir explicarei a relevância de cada uma dessas matéria, algumas das quais podem, a princípio, parecerem irrelevantes para o trabalho realizado. A ordem de apresentação leva em conta minha percepção da importância das disciplinas, ignorando a seqüência cronológica.

6.3.1 Métodos Formais em Programação (MAC 239)

Certamente a disciplina mais importante para o projeto foi Métodos Formais em Programação, visto que através dela aprendi sobre Lógica e comecei a apreciar formalizações em geral. O trabalho realizado é, essencialmente, fruto dessa apreciação. É bem verdade que as demais disciplinas do curso já usavam, implicitamente, os princípios da Lógica (e.g., na prova de teoremas algébricos) e que, afinal, a Matemática toda é uma espécie de formalização. Contudo, tais usos eram restritos a estudos numéricos, os quais, embora essenciais, mascaravam a verdadeira natureza e propósito do pensamento lógico e estruturado – a preservação da *verdade*, seja ela numérica ou não. Foi essa percepção da Lógica enquanto a *melhor* ferramenta analítica de propósito *geral* que me cativou.

6.3.2 Introdução à Inteligência Artificial (MAC 425)

Embora o curso de Introdução à Inteligência Artificial tenha abordado diversos algorítmos importantes (e.g., busca informada), foram os aspectos não algorítmicos, especialmente os de cunho filosófico, que mais influenciaram meu projeto. Em particular, foi mediante essa disciplina que tomei maior contato com o conceito de *agente*, fundamental tanto para a parte técnica quanto para a parte filosófica do meu trabalho.

6.3.3 Conceitos Fundamentais de Linguagens de Programação (MAC 316)

Visto que as linguagens de programação são as principais ferramentas *práticas* na Ciência da Computação, me parece ser de suma importância tomá-las como objetos de estudo. Não só do ponto de vista da implementação, mas também lingüístico, isto é, do ponto de vista expressivo, descritivo. Foi neste último ponto que a disciplina teve especial utilidade com relação ao projeto. Ainda que o método Z seja um método formal, e não uma linguagem de programação, alguns de seus aspectos podem ser estudados e compreendidos sob a luz da teoria para essas linguagens. Por exemplo, conhecendo os diversos paradigmas para linguagens, é fácil ver que Z é *declarativo* e não, por exemplo, imperativo. Uma vantagem em se poder estabelecer tal classificação é que, com ela, traz-se toda teoria já conhecida a respeito de paradigmas declarativos. Ademais, conhecendo-se diversos paradigmas, torna-se mais fácil escolher o correto para a tarefa em mãos.

6.3.4 Algorítmos em Grafos (MAC 328)

Com Algorítmos em Grafos aprendi sobre grafos e suas aplicações. Que grafos são estruturas tecnologicamente úteis, tornou-se rapidamente claro, uma vez que os utilizamos para modelar problemas de interesse prático (e.g., determinar rotas entre cidades) cujas soluções podem ser encontradas por algorítmos eficientes. Porém, o maior valor da matéria só pôde ser observado posteriormente, ao longo do tempo, conforme eu me deparava com problemas redutíveis a problemas de grafos. Em particular, foi muito gratificante quando reconheci uma estrutura de grafo num conjunto de definições psicológicas e o utilizei para formalizá-las no meu projeto.[17]

6.3.5 Laboratório de Programação I e II (MAC 211 e MAC 242)

As disciplinas de Laboratório de Programação propiciaram ótimas experiências no desenvolvimento prático de projetos de *software*. Embora outras matérias exijissem um pouco de programação, em geral esta limitava-se a implementações de algorítmos pequenos, com o propósito de estudar alguma teoria matemática subjacente. Assim, posso afirmar seguramente que os laboratórios tiveram papéis fundamentais tanto nas minhas habilidades como programador quanto na minha vontade de programar. Meu projeto de formatura foi motivado, em grande parte, por essa vontade. Vale ressaltar que, no segundo laboratório, implementamos um autômato celular, isto é, um simulador, no qual busquei um pouco da inspiração necessária para o trabalho de formatura.

6.3.6 Engenharia de Software (MAC 332) e Tópicos de Programação Orientada a Objetos (MAC 413)

Enquanto os laboratórios de programação foram responsáveis pelos aspectos mais práticos da minha educação como desenvolvedor de *software*, as disciplinas de Engenharia de Software e de Tópicos de Programação Orientada a Objetos forneceram os fundamentos teóricos da prática. Os conceitos de arquitetura e *design*, abordados em ambas as disciplinas, foram de grande importância no desenvolvimento do projeto, uma vez que trata-se de um sistema complexo. É interessante notar que mesmo na especificação formal do sistema, que não envolve programação, muitos desses conceitos foram úteis.

6.3.7 Álgebra II (MAT 213)

Álgebra II[18] influenciou profundamente o meu pensamento metodológico, ainda que os objetos de estudo do assunto (i.e., números) não sejam meu interesse principal. A definição de estruturas abstratas, as diversas relações entre tais estruturas e as provas cuidadosas são pontos fortes da disciplina e podem ser

[17] O grafo de estímulos, que define relações causais entre estímulos. Veja a Seção 4.1.

[18] A disciplina chama-se Álgebra II, mas o campo da Matemática por ela estudado leva o nome de Álgebra Abstrata.

facilmente transferidos para outras aplicações além da Álgebra. No tocante ao projeto, esse modo de pensar me fez indagar questões como:

- Será que é possível estabelecer um "isomorfismo"[19] entre o simulador comportamental e animais reais?
- Será que se existisse tal "isomorfismo comportamental", isso bastaria para afirmarmos que os agentes virtuais e os agentes reais são equivalentes?[20]
- Será que as operações sobre os diversos objetos da especificação (e.g., estímulos, comportamentos) estão bem definidas?
- Será que é possível estabelecer classes abstratas de agentes, de modo a facilitar o tratamento de agentes concretos (e.g., animais, robôs)?

6.3.8 Introdução à Computação Gráfica (MAC 420)

Tecnicamente, a matéria de Introdução à Computação Gráfica não desempenhou nenhum papel no projeto. Contudo, do ponto de vista subjetivo, ela teve relevância significativa. O uso da Álgebra Linear e do Cálculo para a geração de imagens *bonitas* teve um impacto razoável na minha apreciação da Matemática. Dentre outras atividades interessantes, desenvolvi um *ray tracer*[21], o qual usei, posteriormente, como ferramenta artísica (veja a Figura 9). Sem uma teoria formal por traz, dificilmente poder-se-ia construir um sistema capaz de gerar os mesmos resultados desse *ray tracer*. Ele evidenciou, portanto, o valor prático das teorias formais.

6.3.9 Programação Linear (MAC 315)

Programação Linear também não contribuiu diretamente para o projeto. No entanto, ela teve importância enquanto exemplo de aplicação da Matemática. Há três aspectos metodológicos que me foram bem úteis:

- O enfoque na procura pela *estrutura* dos problemas;
- A simplificação dos problemas, de modo a torná-los tratáveis;
- A exploração cuidadosa das estruturas simples para alcançar resultados poderosos e não triviais.

[19] Estou utilizando o conceito de isomorfismo informalmente aqui, mas inspirando-me em sua definição formal.

[20] Esta é uma questão pertencente à área da Filosofia da Mente, mas encontra ferramentas úteis nos conceitos da Álgebra Abstrata.

[21] Um ray tracer é um programa que simula a emissão de raios de luz num ambiente virtual, produzindo uma imagem.

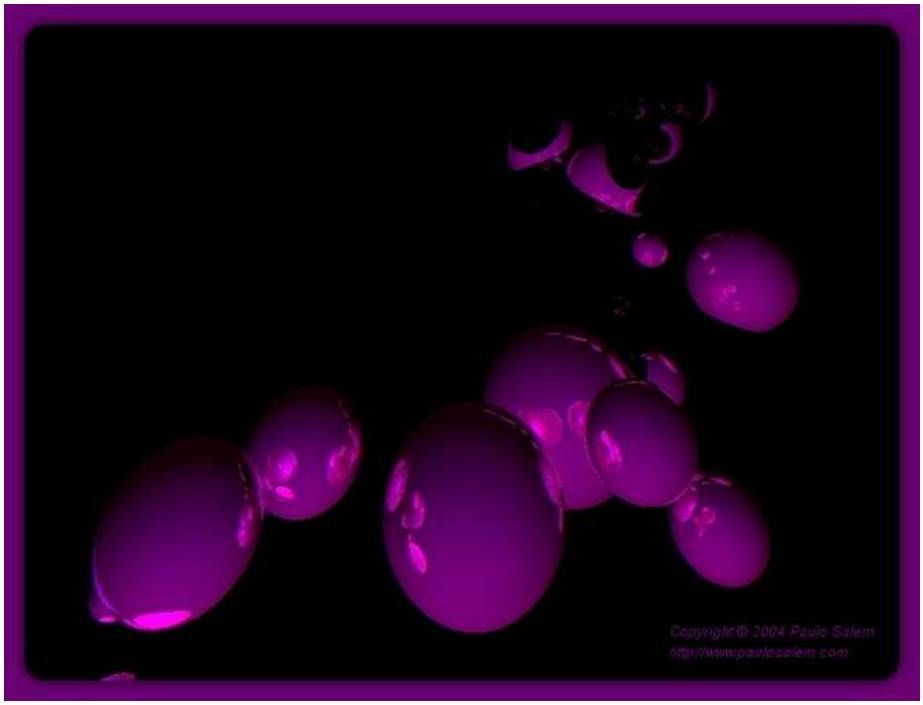

Figura 9: *Purple Bubbles*. Arte abstrata de minha autoria, feita com o *ray tracer* construído por mim e Ellen Hidemi Fukuda durante a disciplina de Introdução à Computação Gráfica. O *ray tracer* gerou as esferas, mas a borda que enquadra a imagem foi acrescida por outro programa.

6.3.10 Introdução aos Fundamentos da Matemática (MAT 350)

A optativa Introdução aos Fundamentos da Matemática melhorou significativamente minha compreensão da Filosofia da Ciência e da Matemática. Novamente, essa influência não teve relação direta com o projeto, mas acredito ter sido muito importante para minha formação científica e, por conseguinte, indiretamente relevante para o meu projeto. Nessa disciplina minha principal atividade foi o estudo de diversos tratados filosóficos, dos quais destaco o livro *The Structure of Scientific Revolutions* [Kuh70] de Thomas Kuhn.

Considero uma lástima o fato desse tipo de conhecimento não fazer parte do currículo obrigatório do BCC, visto que se pretende formar *cientistas* da Computação. Quantos dos graduados do BCC são capazes de discutir a respeito da natureza da Ciência?

6.3.11 Outras Disciplinas

Há disciplinas que, por serem tão fundamentais, possuem importância óbvia tanto para minha formação quanto para meu projeto e, portanto, não merecem discussões à parte. Nessa classe de matérias incluo Introdução à Computação (MAC 110), Álgebra I (MAT 138), Princípios de Desenvolvimento de Algorítmos (MAC 122), Cálculo Diferencial e Integral I e II (MAT 111 e MAT 121, respectivamente) e Estrutura de Dados (MAC 323).

Por fim, enfatizo que considero praticamente todas as demais disciplinas do currículo como valiosas para minha formação geral, embora, em alguns poucos casos, elas tenham sido ministradas de maneiras insatisfatórias. A exceção é Língua Portuguesa (FLC 474), cujo conteúdo me pareceu quase totalmente inútil ainda que razoavelmente bem ministrado. Fundamentalmente, quem não sabia escrever continuou sem saber e quem já sabia não aproveitou muita coisa. Em minha opinião, essa disciplina deveria ser ou eliminada impiedosamente ou reformulada completamente. Minha sugestão e preferência, porém, é que se coloque, no lugar dela, alguma matéria de humanas voltada à Filosofia da Ciência. Desse modo, os alunos teriam a oportunidade tanto de exercitar a língua (e.g., mediante a leitura e escrita de textos), quanto de aprender sobre uma área relevante da Filosofia.

6.4 Iniciação Científica

O projeto de formatura foi feito sob uma iniciação científica. Porém, antes dele eu já havia feito uma outra iniciação, com a mesma orientadora, também na área Métodos Formais. Essencialmente, desenvolvi uma ontologia voltada à verificação formal de agentes móveis. Coloquei esse trabalho na forma de um artigo, o qual foi aceito numa conferência[22] e, por meio dela, publicado pela *Lecture Notes in Computer Science* [SdM05].

Considero essa primeira experiência fundamental para minha formação acadêmica. Ela me deu a oportunidade de me focar num problema real, produzindo um trabalho igualmente real, publicável. Assim, permitiu-me explorar muitas das etapas envolvidas na pesquisa científica verdadeira, indo além, portanto, de um mero exercício curricular.

A orientação da professora Ana Cristina foi de grande valia nesse processo. Não só do ponto de vista técnico, mas também do ponto de vista humano, na medida em que ela sempre se mostrou muito prestativa e compreensiva.

6.5 Atividades Acadêmicas Extra-Curriculares

O valor da USP, particularmente da Cidade Universitária, enquanto espaço de vivência acadêmica é inestimável. Basta lembrar das inúmeras bibliotecas, palestras, eventos e institutos a que tive acesso imediato e constante. Ao longo dos

[22] The 4th International Conference on Ontologies, DataBases, and Applications of Semantics (ODBASE 2005)

anos, eu procurei aproveitar toda essa diversidade. Freqüentei palestras em vários institutos, sobre diversos assuntos. Retirei livros em diversas bibliotecas[23] e visitei inúmeras exposições e apresentações. Até mesmo cursei uma disciplina[24], por mera curiosidade, na Faculdade de Enfermagem. Essa constante imersão em cultura deixou impressões indeléveis no meu espírito.

Com relação ao projeto de formatura, a disponibilidade da biblioteca da faculdade de Psicologia foi essencial. Nela encontrei todo material de que precisei e, em particular, livros antigos e fora de publicação.

6.6 Interação com Pessoas

A grosso modo, posso dizer que tive um bom relacionamento com a maior parte das pessoas com as quais convivi. Os alunos e professores da USP costumam ser pessoas ou agradáveis ou discretas, de modo que é improvável sentir-se incomodado por elas. Ademais, devido ao regime meritocrático da Universidade, é razoavelmente fácil encontrar pessoas com as quais eu consigo conversar. A diferença entre um USPiano e um sujeito qualquer é notável e, para ser sincero, nauseante.

Quanto ao projeto, não há muito o que dizer além do que já expus na seção 6.4. Porém, ressalto novamente que a professora Ana foi uma excelente orientadora e que me incentivou muito no meu progresso acadêmico. Em particular, quando lhe apresentei a proposta do projeto de formatura, ela me apoiou totalmente, não obstante a natureza pouco convencional da minha idéia.

6.7 Planos para o Futuro

Em primeiro lugar, convém ressaltar que meu projeto de formatura não foi começado devido à formatura, tampouco terminará junto com esta. Trata-se de um projeto de valor pessoal, o qual pretendo continuar desenvolvendo até sua conclusão.

Além disso, pretendo continuar meus estudos acadêmicos, ingressando no mestrado em Ciência da Computação do IME. O campo exato em que atuarei ainda está um tanto incerto. A bem da verdade, sou eclético demais para ter certeza do que realmente farei. Todavia, acredito que prosseguirei em alguma linha relacionada aos meus interesses atuais, que se centram em Métodos Formais, Lógica, Inteligência Artificial e Engenharia de Software.

[23] É interessante notar que o salão de leitura da biblioteca do IME é muito mais utilizado do que os salões das outras faculdades que visitei.

[24] Introdução ao Estudo do Stress e do Coping (ENC 160)

A Especificação Parcial

Apresentamos a seguir uma parte considerável da especificação na qual estamos trabalhando. Em seu presente estado, ela encontra-se inconsistente e incompleta, pelos motivos relatados na Seção 6.2. Nosso objetivo ao fornecê-la neste apêndice é meramente mostrar ao leitor a complexidade da formalização. Assim, não provemos texto comentando-na, mas somente a especificação formal em si.

Simulator

$$organism : Organism$$
$$currentInstant : Instant$$

Init

$$Simulator$$

$$Instant == \mathbb{Z}$$
$$Duration == \mathbb{N}$$

$$previousInstants : Instant \rightarrow \mathbb{P}\, Instant$$

$$\forall\, x, y : Instant \bullet y \in previousInstants(x) \Leftrightarrow y < x$$

SimulatorIterationOp

$$\Delta Simulator$$
$$stimulus? : Stimulus$$
$$response! : Response$$

Organism

$$ActionConflict$$
$$stimulusImplication : StimulusImplication$$
$$stimulusEquivalence : StimulusEquivalence$$
$$operantRepertoire : \mathbb{P}\, Operant$$
$$respondentRepertoire : \mathbb{P}\, Reflex$$
$$attention : Attention$$
$$driveSubsystem : DriveSubsystem$$
$$emotionSubsystem : EmotionSubsystem$$
$$stimulationHistory : Instant \nrightarrow (Stimulus \times Intensity)$$

$$SenseOp == Organism\Phi StimulusProcessing^{\circ}_{9}$$
$$\quad Organism\Phi OperantsUpdate^{\circ}_{9}$$
$$\quad Organism\Phi DriveSubsystem^{\circ}_{9}$$
$$\quad Organism\Phi EmotionSubsystem$$

$$RespondOp == Organism\Phi ResponseProcessing$$

$$[Stimulus]$$

$$\mid \nu_s : Stimulus$$

$$StimulusStatus ::= Beginning \mid Ending \mid None$$

$$\mid status_s : Stimulus \rightarrow StimulusStatus$$

Stimulation

$$stimulus : Stimulus$$
$$intensity : Intensity$$
$$status : StimulusStatus$$

StimulationParameters

$$primaryStimuli : \mathbb{P}\, Stimulus$$
$$u_{primary} : \mathbb{P}\, Stimulus \rightarrow Intensity$$
$$max_{delay} : Duration$$

$$\mathrm{dom}\, u_{primary} = primaryStimuli$$

AttentionParameters

Attention

$$saliences : Stimulus$$

StimulusImplication

$$\_ \rightsquigarrow_s \_ : Stimulus \leftrightarrow Stimulus$$
$$intensity : Stimulus \times Stimulus \rightarrow Intensity$$

$$\forall s_1, s_2 : Stimulus \bullet$$
$$\quad s_1 \rightsquigarrow_s s_2 \Leftrightarrow s_1 \rightsquigarrow s_2$$
$$\forall s_1, s_2 : Stimulus \mid s_1 \rightsquigarrow_s s_2 \bullet$$
$$\quad \exists i : Intensity \bullet (s_1 \rightsquigarrow_s s_2 \mapsto i) \in intensity$$

StimulusEquivalence

$$\_ \equiv_s \_ : Stimulus \leftrightarrow Stimulus$$
$$intensity : Stimulus \times Stimulus \rightarrow Intensity$$

$$\forall\, s_1, s_2 : Stimulus \bullet$$
$$\quad s_1 \equiv_s s_2 \Leftrightarrow s_1 \rightsquigarrow_s s_2 \land s_2 \rightsquigarrow_s s_1$$
$$\forall\, s_1, s_2 : Stimulus \mid s_1 \equiv_s s_2 \bullet$$
$$\quad \exists\, i : Intensity \bullet (s_1 \equiv_s s_2 \mapsto i) \in intensity$$

StimulusUtility

$$StimulationParameters$$
$$EmotionSubsystem$$
$$DriveSubsystem$$
$$u_s : Stimulus \rightarrow Intensity$$

$$u_s(\nu_s) = 0$$
$$\exists\, f : Stimulus \times \mathbb{P}\, Emotion \times \mathbb{P}\, Drive \rightarrow Intensity \bullet$$
$$\quad u_s(s) = f(s, activeEmotions, activeDrives)$$

StimulusUtilityBase$_1$

$$StimulusUtility$$
$$StimulusImplication$$
$$base : Stimulus \rightarrow Intensity$$

$$\forall\, s : Stimulus \bullet$$
$$\quad (\exists\, s_p : primaryStimuli \bullet$$
$$\quad\quad base(s) = u_{primary}(s_p) \land$$
$$\quad\quad \forall\, s_q : primaryStimuli \mid s \rightsquigarrow_s s_q \bullet$$
$$\quad\quad\quad u_{primary}(s_p) \geq u_{primary}(s_q) \land$$
$$\quad\quad (s \rightsquigarrow_s s_p)) \lor$$
$$\quad (\forall\, s_p : primaryStimuli \bullet$$
$$\quad\quad s \not\rightsquigarrow_s s_p \land$$
$$\quad\quad u_s(s) = 0)$$

StimulusUtility$_1$

$$StimulusUtilityBase_1$$
$$EmotionalRegulators$$
$$DriveRegulator$$

$$\forall\, s : Stimulus \bullet$$
$$\quad u_s(s) = driveRegulators(s, emotionalRegulators(s, base(s)))$$

ConditioningOp

$\Delta StimulusImplication$

$s_1? : Stimulus$

$s_2? : Stimulus$

$delay? : Duration$

$delay? \leq max_{delay}$

$s_1? \rightsquigarrow'_s s_2?$

$(s_1? \rightsquigarrow_s s_2?) \land intensity(s_1?, s_2?) < 1 \Rightarrow$
$\quad intensity'(s_1?, s_2?) > intensity(s_1?, s_2?)$

$(s_1? \rightsquigarrow_s s_2?) \land intensity(s_1?, s_2?) = 1 \Rightarrow$
$\quad intensity'(s_1?, s_2?) = intensity(s_1?, s_2?)$

$s_1? \not\rightsquigarrow_s s_2? \Rightarrow$
$\quad intensity'(s_1?, s_2?) > 0$

ConditioningDecayOp

$\Delta StimulusImplication$

$s_1? : Stimulus$

$s_2? : Stimulus$

$delay? : Duration$

$delay? > max_{delay}$

$s_1? \rightsquigarrow_s s_2?$

$intensity'(s_1?, s_2?) < intensity(s_1?, s_2?)$

$intensity'(s_1?, s_2?) \leq 0 \Rightarrow$
$\quad s_1? \not\rightsquigarrow'_s s_2?$

$T\_ConditioningOp_1 == ConditioningOp \lor ConditioningDecayOp$

LinearConditioningParameters

$StimulationParameters$

$\alpha_s : Intensity$

$increment : Intensity$

$newIntensity : Intensity$

$delay? : Duration$

$increment = \alpha_s / delay?$

LinearConditioningOp

$StimulusConditioningOp$

$LinearConditioningParameters$

$newIntensity = intensity(s_1?, s_2?) + increment$

$intensity(s_1?, s_2?) + increment \leq max_{intensity} \Rightarrow$
$\quad intensity' = intensity \oplus (s_1?, s_2?, newIntensity)$

$intensity(s_1?, s_2?) + increment > max_{intensity} \Rightarrow$
$\quad intensity' = intensity \oplus (s_1?, s_2?, max_{intensity})$

LinearConditioningDecayOp

$ConditioningDecayOp$

$newIntensity = intensity(s_1?, s_2?) - increment$

$intensity(s_1?, s_2?) - increment \geq 0 \Rightarrow$
$\quad intensity' = intensity \oplus (s_1?, s_2?, newIntensity)$

$intensity(s_1?, s_2?) - increment < 0 \Rightarrow$
$\quad intensity' = intensity \oplus (s_1?, s_2?, 0)$

$T\_ConditioningOp_2 == LinearConditioningOp \lor LinearConditioningDecayOp$

PerceptionOp

$\Delta Organism$

$stimulation? : Stimulus \times Intensity$

$currentInstant? : Instant$

$stimulationHistory' = stimulationHistory \oplus (currentInstant? \mapsto stimulation?)$

OrganismΦStimulusProcessing

$\Delta Organism$

$PerceptionOp$

$stimulation? : \mathbb{P}\, Stimulus \times Intensity$

$currentInstant? : Instant$

$first\ stimulation? \neq \nu_s$

$\forall\, si : stimulationHistory(\!|\ previousInstants(currentInstant)\ |\!) \bullet$
$\quad \exists\, T\_ConditioningOp_1 \bullet$
$\quad \Theta StimulusImplication = \Theta stimulusImplication \land$
$\quad \Theta StimulusImplication' = \Theta stimulusImplication' \land$
$\quad s_1? = first\ stimulation? \land s_2? = first\ si$

$[Action]$

$Conflict ::= conflicting \mid nonconflicting$

ActionConflict

$$conflict : Action \times Action \rightarrow Conflict$$

ActionRegulator

Response

$$action : Action$$
$$latency : Duration$$
$$magnitude : Intensity$$

NextBehaviors

$$elicited : \mathbb{P}\, Reflex$$
$$emitted : \mathbb{P}\, Operant$$

ReflexSchedulingOp

$$\Xi Organism$$
$$\Delta NextBehaviors$$
$$s : Stimulus$$
$$i : Intensity$$

$$\forall r : Reflexes \bullet ReflexElicitationCond \Rightarrow r \in elicited'$$
$$elicited \subseteq elicited'$$

OperantSchedulingOp

$$\Xi Organism$$
$$\Delta NextBehaviors$$

$$\forall o : Operants \bullet OperantEmissionCond \Rightarrow o \in emitted'$$
$$emitted \subseteq emitted'$$

$$ResponseSchedulingOp == ReflexSchedulingOp \land OperantSchedulingOp$$

Reflex

$ReflexParameters$

$antecedent : Stimulus$

$action : Action$

$threshold : Intensity$

$elicitation : Probability$

$magnitude : Intensity$

$duration : Duration$

$latency : Duration$

$elicitation_{min} \leq elicitation \leq elicitation_{max}$

$magnitude_{min} \leq magnitude \leq magnitude_{max}$

$duration_{min} \leq duration \leq duration_{max}$

$latency_{min} \leq latency \leq latency_{max}$

$threshold_{min} \leq threshold \leq threshold_{max}$

ReflexParameters

$\delta_{elicitation} : Probability \times Instant \times Instant \rightarrow Probability$

$\delta_{magnitude} : Intensity \times Instant \times Instant \rightarrow Intensity$

$\delta_{duration} : Intensity \times Instant \times Instant \rightarrow Intensity$

$\delta_{latency} : Duration \times Instant \times Instant \rightarrow Duration$

$\delta_{threshold} : Intensity \times Instant \times Instant \rightarrow Intensity$

$elicitation_{max} : Probability$

$elicitation_{min} : Probability$

$magnitude_{max} : Intensity$

$magnitude_{min} : Intensity$

$duration_{max} : Intensity$

$duration_{min} : Intensity$

$latency_{max} : Duration$

$latency_{min} : Duration$

$threshold_{max} : Intensity$

$threshold_{min} : Intensity$

$$\forall t_1, t_2 : Instant, p : Probability, i_1, i_2 : Intensity, d : Duration \bullet$$
$$elicitation_{min} \leq \delta_{elicitation}(p, t_1, t_2) \leq elicitation_{max} \land$$
$$magnitude_{min} \leq \delta_{magnitude}(i_1, t_1, t_2) \leq magnitude_{max} \land$$
$$duration_{min} \leq \delta_{duration}(i_1, t_1, t_2) \leq duration_{max} \land$$
$$latency_{min} \leq \delta_{latency}(d, t_1, t_2) \leq latency_{max} \land$$
$$threshold_{min} \leq \delta_{threshold}(i_2, t_1, t_2) \leq threshold_{max}$$

ReflexAdjustmentOp

$\Delta Reflex$

$t_1? : Instant$

$t_2? : Instant$

$elicitation' = \delta_{elicitation}(elicitation, t_1?, t_2?)$

$magnitude' = \delta_{magnitude}(magnitude, t_1?, t_2?)$

$duration' = \delta_{duration}(duration, t_1?, t_2?)$

$latency' = \delta_{latency}(latency, t_1?, t_2?)$

$\theta ReflexParameters' = \theta ReflexParameters$

$antecedent' = antecedent$

$action' = action$

ReflexElicitationCond

$$
\begin{array}{l}
StimulusImplication \\
r : Reflex \\
s : Stimulus \\
i : Intensity \\
\hline
s \rightsquigarrow_s (r.antecedent) \\
i \geq (r.threshold)
\end{array}
$$

OrganismΦ*ReflexAdjustment*

$$
\begin{array}{l}
\\
\hline
\\
\hline
\end{array}
$$

OperantSubsystemParameters

$$
\begin{array}{l}
\\
\hline
\\
\hline
\end{array}
$$

DiferentiableProperties

$$
\begin{array}{l}
interResponseTime : Duration \\
\hline
\\
\hline
\end{array}
$$

Operant

$$
\begin{array}{l}
antecedents : \mathbb{P}\,\mathbb{P}\,Stimulus \\
action : Action \\
consequence : Stimulus \\
consequenceContingency : (\mathbb{P}\,Stimulus) \nrightarrow Intensity \\
stimulationHistory : Instant \nrightarrow (Stimulus \times Intensity) \\
responseHistory : Instant \nrightarrow Response \\
\hline
consequence \neq \nu_s \\
\varnothing \in antecedents \\
\mathrm{dom}\, consequenceContingency = antecedents
\end{array}
$$

OperantImplication

$$
\begin{array}{l}
StimulusImplication \\
\_ \rightsquigarrow_o \_ : (Operant \cup Stimulus) \leftrightarrow (Operant \cup Stimulus) \\
implicationIntensity : (Operant \cup Stimulus) \times (Operant \cup Stimulus) \nrightarrow Intensity \\
\hline
\forall\, x, y : (Operant \cup Stimulus) \bullet \\
\quad x \rightsquigarrow_o y \Leftrightarrow x \rightsquigarrow y \\
\forall\, o : Operant, s : Stimulus \bullet \\
\quad s \rightsquigarrow_o o \Rightarrow \exists\, s' : Stimulus \bullet o \rightsquigarrow_o s' \\
\forall\, o_1, o_2 : Operant \bullet \\
\quad o_1 \not\rightsquigarrow_o o_2 \\
\forall\, x, y : (Operant \cup Stimulus) \mid x \rightsquigarrow_o y \bullet \\
\quad \exists\, i : Intensity \bullet ((x \rightsquigarrow_o y) \mapsto i) \in implicationIntensity
\end{array}
$$

OperantUtility

$$
\begin{array}{l}
StimulusUtility_1 \\
OperantImplication \\
u_o : (Operant \times \mathbb{P}\, Stimulus) \rightarrow Intensity \\
\hline
\\
\end{array}
$$

OperantUtility$_1$

$$
\begin{array}{l}
OperantUtility \\
\hline
\forall\, o : Operant, S : \mathbb{P}\, Stimulus \bullet \\
\quad \exists\, s : Stimulus, s_1 : S \mid o \rightsquigarrow_o s \lor (s_1 \rightsquigarrow_o o \land o \rightsquigarrow_o s) \bullet \\
\qquad u_o(o, S) = u_s(s) \land \\
\qquad \forall\, s' : Stimulus, s_2 : S \mid o \rightsquigarrow_o s' \lor (s_2 \rightsquigarrow_o o \land o \rightsquigarrow_o s') \bullet \\
\qquad\quad u_s(s) \geq u_s(s')
\end{array}
$$

OperantOp

$\Delta Operant$
$\Delta EmotionalSubsystem$
$OperantUtility$
$StimulusImplication$
$discriminativeStimuli? : \mathbb{P}\, Stimulus$
$consequence? : Stimulus$
$response? : Action$
$delay? : Duration$

$response? = action$
$consequence? \rightsquigarrow_s consequence$
$delay? \leq max_{delay}$

OperantFormationOp

$\Delta EmotionalSubsystem$
$action? : Action$
$consequence? : Stimulus$
$discriminativeStimuli? : \mathbb{P}\, Stimulus$
$delay? : Duration$
$new! : Operant$

$delay? < max_{delay}$
$discriminativeStimuli? \in new!.antecedents$
$\forall\, S_d : \mathbb{P}\, Stimulus \bullet S_d \in new!.antecedents \Leftrightarrow S_s = discriminativeStimuli?$
$new!.consequence = consequence?$
$new!.action = action?$
$domnew!.consequenceContingency = \{discriminativeStimuli?\}$
$\mathrm{dom}\, new!.stimulationHistory = \varnothing$
$\mathrm{dom}\, new!.responseHistory = \varnothing$

DiscriminationOp

$OperantOp$

$discriminativeStimuli? \notin \mathrm{dom}\, consequenceContingency$
$discriminativeStimuli? \in \mathrm{dom}\, consequenceContingency'$
$consequenceContingency'(discriminativeStimuli?) > 0$

OperantConditioningOp

$$OperantOp$$

$$discriminativeStimuli? \in \mathrm{dom}\ consequenceContingency$$
$$consequence? \rightsquigarrow_s consequence$$
$$consequenceContingency'(discriminativeStimuli?) > consequenceContingency(discriminativeStimuli?)$$

ExtinctionOp

$$OperantOp$$

$$discriminativeStimuli? \in \mathrm{dom}\ consequenceContingency$$
$$consequence? \not\rightsquigarrow_s consequence$$
$$consequenceContingency'(discriminativeStimuli?) < consequenceContingency(discriminativeStimuli?)$$

$$T\_OperantConditioningOp == DiscriminationOp \lor OperantConditioningOp \lor ExctinctionOp$$

IRTDiferentiationOp

$$OperantOp$$

ResponseRateDiferentiationOp

$$OperantOp$$

$$DiferentiationOp == IRTDiferentiationOp \land ResponseRateDiferentiationOp$$

Reinforcement

$$StimulusUtility_1$$
$$consequence? : Stimulus$$

$$u_s(consequence?) > 0$$

$$ReinforcementOp_1 == OperantOp \land Reinforcement \land T\_OperantConditioningOp$$
$$ReinforcementOp_2 == OperantFormationOp \land Reinforcement$$

PositiveReinforcement

$$StimulusUtility_1$$
$$consequence? : Stimulus$$

$$u_s(consequence?) > 0$$
$$status_s(consequence?) = Beginning$$

$$PositiveReinforcementOp_1 == ReinforcementOp_1 \land PositiveReinforcement$$
$$PositiveReinforcementOp_2 == ReinforcementOp_2 \land PositiveReinforcement$$

NegativeReinforcement

$$StimulusUtility_1$$
$$consequence? : Stimulus$$

$$u_s(consequence?) > 0$$
$$status_s(consequence?) = Ending$$

$$NegativeReinforcementOp_1 == ReinforcementOp_1 \land NegativeReinforcement$$
$$NegativeReinforcementOp_2 == ReinforcementOp_2 \land NegativeReinforcement$$

Punishment

$$StimulusUtility_1$$
$$consequence? : Stimulus$$

$$u_s(consequence?) < 0$$

$$PunishmentOp_1 == OperantOp \land Punishment \land T\_OperantConditioningOp$$
$$PunishmentOp_1 == OperantFormationOp \land Punishment$$

PositivePunishment

$$StimulusUtility_1$$
$$consequence? : Stimulus$$
$$status_s(consequence?) = Beginning$$

$$PositivePunishmentOp_1 == PunishmentOp_1 \land PositivePunishment$$
$$PositivePunishmentOp_2 == PunishmentOp_2 \land PositivePunishment$$

NegativePunishment

$$StimulusUtility_1$$
$$consequence? : Stimulus$$
$$status_s(consequence?) = Ending$$

$$NegativePunishmentOp_1 == PunishmentOp_1 \land NegativePunishment$$
$$NegativePunishmentOp_2 == PunishmentOp_2 \land NegativePunishment$$

NeutralOperantOp$_1$

$$OperantOp$$
$$u_s(consequence?) = 0$$

NeutralOperantOp$_2$

$$OperantFormationOp$$
$$u_s(consequence?) = 0$$

$$T\_OperantOp == PositiveReinforcementOp_1 \lor NegativeReinforcementOp_1 \lor PositivePunishmentOp_1 \lor NegativePunishmentOp_1 \lor NeutralOperantOp_1$$
$$T\_OperantFormationOp == PositiveReinforcementOp_2 \lor NegativeReinforcementOp_2 \lor PositivePunishmentOp_2 \lor NegativePunishmentOp_2 \lor NeutralOperantOp_2$$

OperandEmissionCond

$$OperantUtility$$
$$allOperants : \mathbb{P}\ Operant$$
$$o : Operant$$
$$\forall\ o' : Operant \bullet u_o(o) \geq u_o(o')$$

DriveSubsystem

$$drives : Drive$$
$$\forall\ d1 : drives, d2 : drives \bullet d1 = d2 \Leftrightarrow d1.name = d2.name$$

Drive

$$name : Name$$
$$level : Intensity$$
$$naturalChange : Intensity \rightarrow Intensity$$
$$desires : \mathbb{P}\ StimulusProperty$$

EmotionParameters

$$intensity : Intensity$$
$$duration : Duration$$

Anger$_1$

$$EmotionParameters$$
$$targets : \mathbb{P}\ Stimulus$$

Anxiety$_1$

$$EmotionParameters$$

Aversion$_1$

$$EmotionParameters$$
$$targets : \mathbb{P}\ Stimulus$$

Depression$_1$

$$EmotionParameters$$

Fear$_1$

EmotionParameters

Frustration$_1$

EmotionParameters

Joy$_1$

EmotionParameters

Relief$_1$

EmotionParameters

$$Emotion == Anger_1 \cup Anxiety_1 \cup Aversion_1 \cup Depression_1 \cup Fear_1 \cup Frustration_1 \cup Joy_1 \cup Relief_1$$

EmotionSubsystem

StandardEmotions

$activeEmotions : \mathbb{P}\, Emotion$

StimulusEmotionalRegulators

DepressionRegulator

$emotionalRegulators : (Stimulus \times Intensity) \rightarrow Intensity$

$$\forall\, s : Stimulus, i : Intensity \bullet emotionalRegulators(s, i) = depression(s, i)$$

DepressionRegulator$_1$

EmotionSubsystem

$depression : (Stimulus \times Intensity) \rightarrow Intensity$

$$\exists\, d : Depression_1 \bullet d \in activeEmotions$$

$$\forall\, s : Stimulus, i : Intensity \mid d.intensity * i \neq 0 \bullet depression(s, i) = i - (i/(d.intensity * i))$$

$$\forall\, s : Stimulus, i : Intensity \mid d.intensity * i = 0 \bullet depression(s, i) = 0$$

$$
\begin{array}{|l}
\hline
DepressionRegulator_2 \\
EmotionSubsystem \\
depression : (Stimulus \times Intensity) \rightarrow Intensity \\
\hline
\nexists\, d : Depression_1 \bullet d \in activeEmotions \\
\forall\, s : Stimulus, i : Intensity \bullet \\
\quad depression(s, i) = i \\
\hline
\end{array}
$$

$$DepressionRegulator == DepressionRegulator_1 \lor DepressionRegulator_2$$

$$
\begin{array}{|l}
\hline
EmotionInjectionOp \\
\Delta EmotionSubsystem \\
\hline
\end{array}
$$

B Ontologia Inicial

Mostramos a seguir alguns diagramas da ontologia produzida no início do projeto, cujo propósito foi o estudo dos conceitos da Análise do Comportamento segundo as definições de [Ski53]. Essa ontologia, porém, encontra-se desatualizada e não reflete nossos conhecimentos atuais da área. A mostramos aqui enquanto artefato histórico. Porém, acreditamos que ela possa ser modificada de modo a ser útil não só como artifício educacional, mas também como recurso tecnológico.

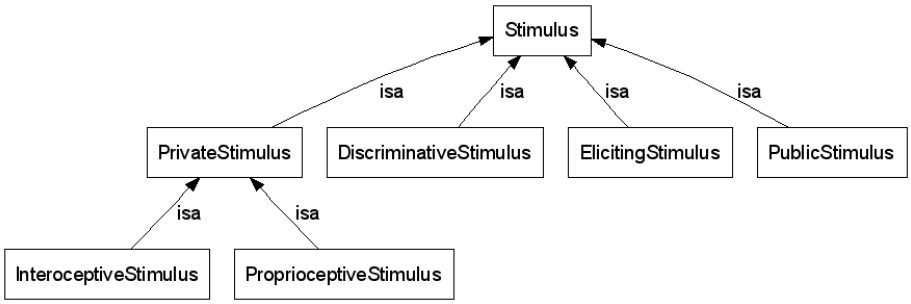

Figura 10: Alguns tipos de estímulo.

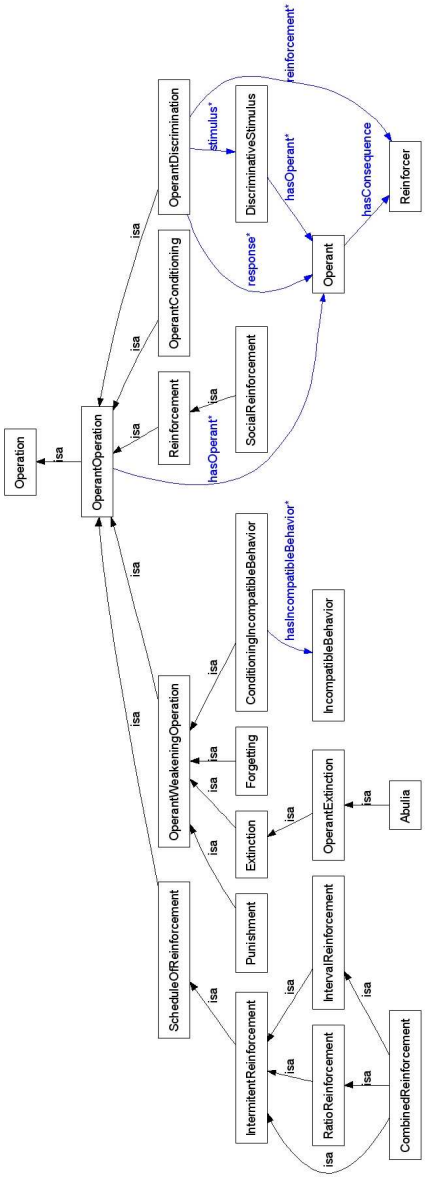

Figura 11: Operações sobre operantes.

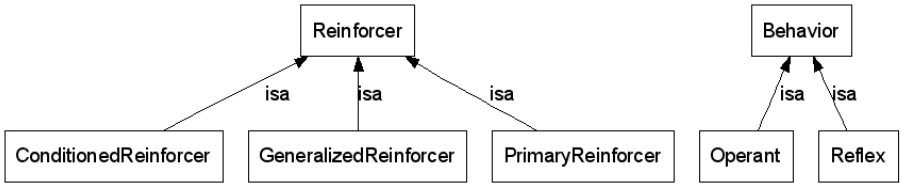

Figura 12: Comportamentos e reforços.

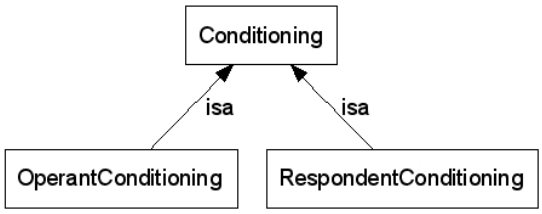

Figura 13: Tipos de condicionamento.

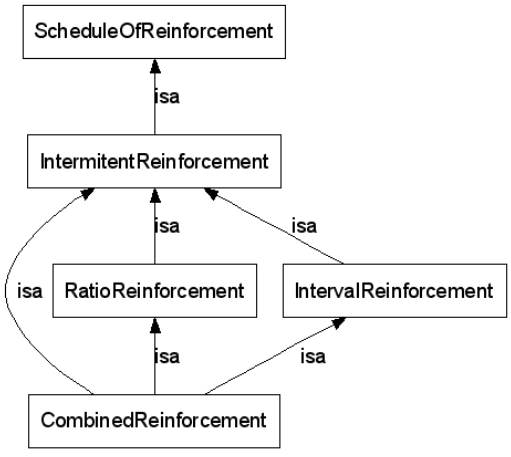

Figura 14: Escalonamentos de reforço.

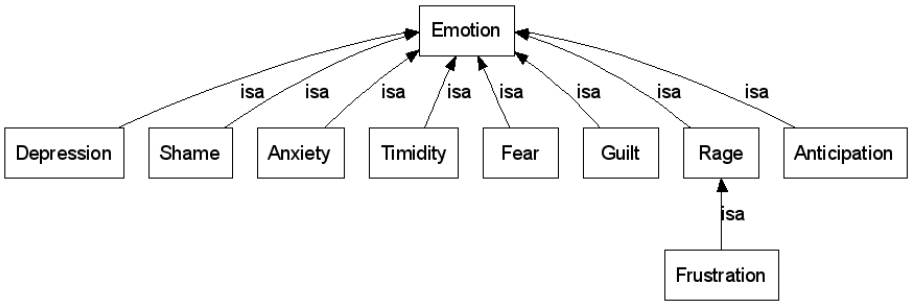

Figura 15: Emoções.

Referências

[Bow] Jonathan Bowen. Z user group. http://www.zuser.org/.

[Cam03] Albert Camus. *A Peste*. Record, 2003. Tradução de Valerie Rumjanek.

[Cat98] Charles A. Catania. *Learning*. Prentice Hall, 1998.

[FDA03] FDA. Fda seeks injunction against multidata systems intl., 5 2003. http://www.fda.gov/bbs/topics/NEWS/2003/NEW00903.html.

[GG95] Nicola Guarino and Pierdaniele Giaretta. Ontologies and knowledge bases: Towards a terminological clarification. In *Towards Very Large Knowledge Bases: Knowledge Building and Knowledge Sharing*, pages 25–32. IOS Press, 1995.

[Gru93] Thomas R. Gruber. A translation approach to portable ontologies. *Knowledge Acquisition*, 5:199–220, 1993.

[HVV03] Karen Huffman, Mark Vernoy, and Judith Vernoy. *Psicologia*. Atlas, 2003. Coordenação da tradução de Maria Emilia Yamamoto e revisão técnica de Agostinho Minicucci.

[ISO02] ISO. *ISO/IEC 13568:2002 Information technology – Z formal specification notation – Syntax, type system and semantics*. International Standards Organization, 2002.

[Jac96a] Jonathan Jacky. Safety-critical computing: Hazards, practices, standards, and regulation. In Rob Kling, editor, *Computerization and Controversy: Value Conflicts and Social Choices*, pages 767–792. Academic Press, second edition, 1996.

[Jac96b] Jonathan Jacky. *The way of Z: practical programming with formal methods*. Cambridge University Press, New York, NY, USA, 1996.

[Kuh70] Thomas S. Kuhn. *The Structure of Scientific Revolutions*, volume 2 of *International Encyclopedia of Unified Science*. The University of Chicago Press, 1970.

[Lam94] Leslie Lamport. *LaTeX: A Document Preparation System*. Addison-Wesley, 1994.

[Pro] The protégé ontology editor and knowledge acquisition system. http://protege.stanford.edu/.

[RN03] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, 2003.

[SdM05] Paulo Salem and Ana C. V. de Melo. An ontology for mobile agents in the context of formal verification. In *On the Move to Meaningful Internet Systems 2005: CoopIS, DOA, and ODBASE: OTM Confederated International Conferences, CoopIS, DOA, and ODBASE 2005, Agia Napa, Cyprus, October 31 - November 4, 2005, Proceedings Part II*, volume 3761 of *Lecture Notes in Computer Science*, pages 1500–1516. Springer-Verlag, 2005.

[Ski53] Burrhus Frederic Skinner. *Science and Human Behavior*. The Free Press, 1953.

[Spi92] J. Mike Spivey. *The Z Notation: a reference manual*. Prentice Hall, second edition, 1992.

[W3Ca] W3C. W3c semantic web. http://www.w3.org/2001/sw/.

[W3Cb] W3C. Web ontology language. http://www.w3.org/2004/OWL/.

[Wat13] John B. Watson. Psychology as the behaviorist views it. *Psychological Review*, (20):158–177, 1913.

[WD96] Jim Woodcock and Jim Davies. *Using Z: Specification, Refinement, and Proof*. Prentice Hall, 1996.